Azure SQL DB Migration Wizard

Azure SQL DB Migration Wizard (SQLAzureMW) is a tool that was developed to help a DBA migrate their SQL Server 2005 (and later) database to Azure SQL DB. This document covers what you need to know to run this tool.

Author: George Huey
Sept 9th, 2013
Version: 3.11, 4.11, 5.11

Table of Contents

Introduction

In the spring of 2009, Microsoft introduced SQL Azure (now called Azure SQL Database) which is database service built upon SQL Server technology. In the early stages of Azure SQL Database, there was no easy way for users to migrate their on-premise SQL Server database to Azure SQL Database. In order to ease migration of on- premise SQL Server database to Azure SQL Database a tool called SQLAzureMW was developed.

This paper provides an overview of SQLAzureMW and describes how you can use it to migrate your SQL Server 2005 (or later) database to Azure SQL Database.

Compatibility

There are a lot of compatibility issues that you need to take into consideration before migrating your on premise SQL database to Azure SQL Database. Azure SQL Database is constantly being upgraded to run on the latest SQL Server technology (i.e. SQL Server 2014). This means that if you want to migrate your SQL Server 2000 database or your SQL Server 2005 database to Azure SQL Database, you need to make sure that your databases are compatible with SQL Server 2014. For example, earlier versions of SQL Server support the definition of outer joins that use the *= and =* operators in the WHERE clause. This provides join operators in the FROM clause. When the compatibility level of the database is set to SQL Server 2005 or SQL Server 2008, the outer join operators (*= and =*) are not supported. This is only one example of compatibility issues that you will find from migrating older SQL Server databases to the latest SQL Server and Azure SQL Database technologies. It is not the purpose of this whitepaper to discuss in detail the upgrade process from older versions of SQL Server to SQL Server 2014. If you are interested in best practices around upgrading your SQL Server database to SQL Server 2014 please check out "Upgrade to SQL Server 2014".

You will find that the best migration experience for migrating to Azure SQL Database is migrating from a SQL Server 2008 R2 (or later) compatible database. This means that if you want to migrate your SQL Server 2000 / 2005 database to Azure SQL Database, then you need to upgrade your database to SQL Server 2008 R2 (or later) before you migrate to Azure SQL Database. Once you verified that your database is compatible with SQL Server 2014, you can migrate your SQL database to Azure SQL Database.

Now, that said, Azure SQL Database does not support 100% of latest SQL Server functionality (see: Azure SQL Database Transact-SQL Reference). For example, fulltext, rowguid and filestream are not currently supported in Azure SQL Database. SQLAzureMW will do its best to identify compatibility issues, fix them where possible and notify off all issues that it knows about.

Azure SQL Database Migration Wizard (SQLAzureMW)

SQLAzureMW is a tool that will help you migrate your SQL Server 2005 (or later) database to Azure SQL Database. It looks for patterns that have been identified as incompatible and flags incompatibility issues and fixes them when possible. Also note that SQLAzureMW uses Regex and does a pattern search. It is not a full TSQL parser and can thus miss something or report something that is invalid when it is not. For example: if you have a variable declared in a stored procedure named @rowguid, SQLAzureMW would find rowguid and report that it is not supported. While that is true, the use of as a variable name is not. So, when you look through the analysis, be sure to look at the warning and validate that the TSQL flagged. If it look ok to you, then ignore the warning and let it run against Azure SQL DB.

NotSupportedByAzureFile.Config

The key to SQLAzureMW is a file called NotSupportedByAzureFile.Config.

```
<?xml version="1.0"?>
<TSQLNotSupportedByAzure xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <DefaultMessage>{0} is not supported in current version of SQL Azure</DefaultMessage>
    <Skip>
        <SupportedStatement Text="SET ANSI_NULLS ON" />
        <SupportedStatement Text="SET QUOTED_IDENTIFIER ON" />
    </Skip>
    <Table>
        <TableStatement>
            <NotSupported Text="NOT\s[FOR\s]*REPLICATION" ReplaceWith="" SeverityLevel="0" ReplaceString="true" DisplayWarning="true" DefaultMessage="false
            <NotSupported Text="\sON\s\[a-z0-9\]" ReplaceWith="" SeverityLevel="0" ReplaceString="true" DisplayWarning="false" DefaultMessage="false" Warni
            <NotSupported Text="WITH FILLFACTOR\s*=\s*[0-9]*" ReplaceWith="" SeverityLevel="0" ReplaceString="true" DisplayWarning="false" DefaultMessage="
            <NotSupported Text="\(CONTENT \[.*\]\)" ReplaceWith="" SeverityLevel="0" ReplaceString="true" DisplayWarning="true" DefaultMessage="false" Warn
            <NotSupported Text="\]\sROWGUIDCOL" ReplaceWith="]" SeverityLevel="0" ReplaceString="true" DisplayWarning="true" DefaultMessage="false" Warning
            <NotSupported Text="newsequentialid" ReplaceWith="newid" SeverityLevel="1" ReplaceString="true" DisplayWarning="true" DefaultMessage="false" Wa
            <NotSupported Text="SUSER_NAME\(" ReplaceWith="" SeverityLevel="2" ReplaceString="false" DisplayWarning="true" DefaultMessage="true" WarningMes
            <NotSupported Text="\s##" ReplaceWith="" SeverityLevel="2" ReplaceString="false" DisplayWarning="true" DefaultMessage="false" WarningMessage="G
            <NotSupported Text="LOCK_ESCALATION" ReplaceWith="" SeverityLevel="2" ReplaceString="false" DisplayWarning="true" DefaultMessage="true" Warning
            <NotSupported Text="\sSWITCH\sPARTITION\s" ReplaceWith="" SeverityLevel="2" ReplaceString="false" DisplayWarning="true" DefaultMessage="true" W
            <NotSupported Text="ALTER\sPARTITION\s" ReplaceWith="" SeverityLevel="2" ReplaceString="false" DisplayWarning="true" DefaultMessage="true" Warn
        </TableStatement>
    </Table>
    <Index>
        <IndexOptions>
            <NotSupported Text="([,\s]|\()+(ALLOW_(ROW|PAGE)_LOCKS|PAD_INDEX)\s*=\s*(ON|OFF)" ReplaceWith="" SeverityLevel="0" ReplaceString="true" Display
            <NotSupported Text="\)\sON\s\[.*\]\)?" ReplaceWith=")" SeverityLevel="0" ReplaceString="true" DisplayWarning="false" DefaultMessage="false" War
            <NotSupported Text="CREATE\s(PRIMARY)?\s?(FULLTEXT|XML)\sINDEX" ReplaceWith="" SeverityLevel="2" ReplaceString="false" DisplayWarning="true" De
            <NotSupported Text="([,\s]|\()+FILLFACTOR\s*=\s*[0-9]*" ReplaceWith="" SeverityLevel="0" ReplaceString="true" DisplayWarning="false" DefaultMes
            <NotSupported Text="([,\s]|\()+MAXDOP\s*=\s*[0-9]*" ReplaceWith="" SeverityLevel="0" ReplaceString="true" DisplayWarning="false" DefaultMessage
            <NotSupported Text="([,\s]|\()+SORT_IN_TEMPDB\s*=\s*(ON|OFF)" ReplaceWith="" SeverityLevel="0" ReplaceString="true" DisplayWarning="false" Defa
        </IndexOptions>
    </Index>
    <Schema>
        <SchemaChecks>
            <NotSupported Text="NT\sAUTHORITY\\NETWORK\sSERVICE" ReplaceWith="" SeverityLevel="2" ReplaceString="false" DisplayWarning="true" DefaultMessag
        </SchemaChecks>
    </Schema>
    <View>
        <ViewStatement>
        </ViewStatement>
    </View>
    <GeneralTSQL>
        <NotSupported Text="\w+\s?\(\s?(FASTFIRSTROW[,\s]*|FORCESEEK[,\s]*|HOLDLOCK[,\s]*|NOLOCK[,\s]*|NOWAIT[,\s]*|PAGLOCK[,\s]*|READCOMMITTED[,\s]*|READC
        <NotSupported Text="ALTER\sPARTITION\s" ReplaceWith="" SeverityLevel="2" ReplaceString="false" DisplayWarning="true" DefaultMessage="true" WarningM
```

Figure 1: NotSupportedByAzureFile.Config

Currently, NotSupportedByAzureFIle.config has 7 major areas and lots of options groups.

1) **DefaultMessage** – This is the default message that will be displayed. Note that the “Text” in the NotSupported nodes is Regex. When SQLAzureMW finds a match, the match value will be displayed in the {0} area. So, if you don’t want to use the default message, you can create your own message and include the {0} in your message so that you can see what the match actually was.
2) **Skip** – Entries in the <Skip> section applies to any string that does not need to go through the compatibility checks.
3) **Table** – Entries in the <TableStatement> section applies to table Data Definition Language (DDL) statements.
4) **Schema** – Entries in the <Schema> section applies only to schemas.
5) **View** – Entries in the <ViewStatement> section applies only to views
6) **Index** – Entries in the <IndexOptions> section applies only to index options on CREATE and ALTER DDL.
7) **GeneralTSQL** – Entries in the <GeneralTSQL> section applies to all other types of Data Manipulation Language (DML) such as what is found inside of stored procedures.
8) **Other Options**: – At this time, to see what is in each group, find the groups in the config file and look at the Regex expressions in the ‘Text’ field.

a. ActiveDirectorySP
b. BackupandRestoreTable
c. ChangeDataCapture
d. CompabilityChecks
e. CursorSPs
f. DatabaseEngineSP
g. DatabaseMailSP
h. DatabaseMaintenancePlan
i. DataControl
j. DistributedQueriesSP
k. FullTextSearchSP
l. GeneralExtendedSPs
m. IntegrationServicesTable
n. LogShipping
o. MetadataFunction
p. OLEAutomationSP
q. OLEDBTable
r. ProfilerSP
s. ReplicationSP
t. ReplicationTable
u. RowsetFunction
v. SecurityFunction
w. SecuritySP
x. SQLMailSP
y. SQLServerAgentSP
z. SQLServerAgentTable
aa. SystemCatalogView
bb. SystemFunction
cc. SystemStatisticalFunction
dd. Unclassified

As you can see in Figure 1 above, an entry is a node called NotSupported. NotSupported is made up of the following items:

1) **Text** – This is the pattern that you want SQLAzureMW to look for. This pattern is a Regex expression. One site that you can go to too learn more about Regex expressions is: http://www.regexlib.com/.
2) **SearchReplace** – Once the pattern described the "Text" field has been found, SearchReplace is used to perform a secondary search REGEX within that string where you can search and replace something within that string that matches that REGEX pattern. For example, you have a query that uses TSQL hints without the "WITH" clause and you need to find all select statements that have a hint without a WITH clause and insert the WITH before the hint. You can do this by telling SearchReplace to find the "(" and replace it with "WITH (" (See ReplaceWith). Note that since SearchReplace is a REGEX search, you will need to use a REGEX pattern. So in this case it would be SearchReplace = "\(".

   *** NOTE *** When you specify SearchReplace, you will be doing a substring search / replace within the text found in the original pattern. If you wish to replace the contents of the original string found (no sub search), then leave SearchReplace empty or remove it from the XML statement.

Here is an example of one of the rules in SQLAzureMW:

```
<NotSupported
Text="\w+\s?\(\s?(FASTFIRSTROW[,\s]*|FORCESEEK[,\s]*|HOLDLOCK[,\s]*|NOLOCK[,\s]*|NOWAIT[,\s]*|PAGLOCK[,\s
]*|READCOMMITTED[,\s]*|READCOMMITTEDLOCK[,\s]*|READPAST[,\s]*|READUNCOMMITTED[,\s]*|REPEATABLEREAD[,\s]*|
ROWLOCK[,\s]*|SERIALIZABLE[,\s]*|TABLOCK[,\s]*|TABLOCKX[,\s]*|UPDLOCK[,\s]*|XLOCK[,\s]*)+\s?\)"
SearchReplace="\(" ReplaceWith="WITH (" NotSubStr="WITH" SeverityLevel="1" ReplaceString="true"
DisplayWarning="true" DefaultMessage="false" WarningMessage="Deprecated feature 'Table hint without
WITH'.  Automatically added WITH for you." />
```

The above rule will find any select statement with one (or many) of the valid TSQL hints, verify that the WITH statement does not already exist and then insert "WITH (" into the statement.

3) **ReplaceWith** – Once the pattern described the "Text" field has been found, ReplaceWith is what SQLAzureMW will replace that pattern with.
4) **SeverityLevel** – Specifies the color of the message being displayed in the analysis results window. There are three levels of severity:
    a. **0** – This is the lowest level or information level message. The message will be displayed in black.
    b. **1** – This is a medium level or warning level message. This message will be displayed in brown.
    c. **2** – This is a high level or error message that requires action on the users' part. This message will be displayed in red.
5) **ReplaceString** – This field (true or false) indicates whether you want to replace the pattern defined in the "Text" field with the contents in the ReplaceWith field.
6) **NotSubStr** – This field is used make sure that the string found by the Regex Text expressions does not contain a specific sub string. Stated another way, I want a string that contains xxx and zzz but not ddd.
7) **DisplayWarning** – This field (true or false) indicates whether you want a message displayed or not when the pattern defined in the "Text" field is found.
8) **DefaultMessage** – This field tells SQLAzureMW to use the default error message.
9) **WarningMessage** – This is the message that is displayed in the analysis results window when the pattern defined in the "Text" field is found.

The entries within the NotSupportedByAzureFile.config are not 100% inclusive. If you run the SQLAzureMW and find a pattern of errors in the TSQL that is generated, you can add an entry in the config file that looks for that pattern and removes it. Note that sometimes, you will find that one of the entries in the config file is causing you problems. You can either modify the Regex expression or just remove the entry.

App.config (SQLAzureMW.exe.config)

The second most important file is App.config. Here are some of the main entries that you will be of interest to you:

1) **Source Server (Can be SQL Server or Azure SQL DB)**
    a. **SourceConnectNTAuth –** You set this value to "true" for a trusted (Windows Auth) connection. When set to true, username and password will not be used.
    b. **SourceServerName** – Enter the name of your source SQL Server (either SQL Server or Azure SQL DB). In the example below the source server is on the local machine. You can point to any SQL Server that you have access to either on-premise or in Azure SQL DB.
    c. **SourceUserName** – This is your user name that you use to connect to SQL Server / Azure with if you are not doing trusted authentication**.**
    d. **SourcePassword** – This is your password that you use to connect to SQL Server (or Azure SQL DB). Note, that if you don't want to enter your password in the App.config file, you can enter it at the stage where you are ready to connect to SQL Server / Azure in SQLAzureMW.
    e. **SourceDatabase** – If you specify a source database, then only that database will be displayed in the database list.
2) **Target Server (Can be SQL Server or Azure SQL DB)**
    a. **TargetConnectNTAuth –** You set this value to "true" for a trusted (Windows Auth) connection. When set to true, username and password will not be used.

b. **TargetServerName** – Enter the name of your source SQL Server (either SQL Server or Azure SQL DB). In the example below the source server is on the local machine. You can point to any SQL Server that you have access to either on-premise or in Azure SQL DB.
c. **TargetUserName** – This is your user name that you use to connect to SQL Server / Azure with if you are not doing trusted authentication**.**
d. **TargetPassword** – This is your password that you use to connect to SQL Server (or Azure SQL DB). Note, that if you don't want to enter your password in the App.config file, you can enter it at the stage where you are ready to connect to SQL Server / Azure in SQLAzureMW.
e. **TargetDatabase** – If you specify a target database, that database must already exist. When connected to the target database, you will not be able to work (create or delete) any other databases.
f. **TargetServer** – Use this key to specify your target server. The two valid values are:

1. **TargetServerSQLAzure** --> Azure SQL DB
2. **TargetServerSQLServer** --> SQL Server (no analysis is done)

*** Note ***

When your source or target is Azure SQL DB:

- Your SourceConnectNTAuth or TargetConnectNTAuth will need to be set to false since Azure SQL DB does not do Windows (trusted) Authentication.
- Azure SQL DB server name looks something like this: tq61zf7bn.ctp.database.windows.net. You can get your Azure SQL DB server name from the Azure SQL DB web site.
- **When you specify the user name, you MUST have @server as part of the user name or BCP fill fail**. For example: SAUser@tq61zf7bn

```
<appSettings>
  <add key="SourceConnectNTAuth" value="true"/>
  <add key="SourceServerName" value="localhost" />
  <add key="SourceUserName" value="" />
  <add key="SourcePassword" value="" />
  <add key="SourceDatabase" value="" />

  <!-- NOTE BCP requires that you have the SQL Azure server name the end of the user name ie. UserSA@jy
  <add key="TargetConnectNTAuth" value="false"/>
  <add key="TargetServerName" value="SERVER.database.windows.net"/>
  <add key="TargetUserName" value="UserName@SERVER"/>
  <add key="TargetPassword" value=""/>
  <add key="TargetDatabase" value="" />

  <add key="TargetServer" value="TargetServerSQLAzure"/>            <!-- TargetServerSQLAzure = "SQL A
                                                                         TargetServerSQLAzureFed = "SQ
                                                                         TargetServerSQLServer = "SQL

  <add key="ConnectionStringTrusted" value="server={0};database={1};Trusted_Connection=yes;Encrypt=true
  <add key="ConnectionString" value="server={0};database={1};uid={2};pwd={3};Encrypt=true;TrustServerCe

  <add key="BCPExe" value="bcp.exe"/>                               <!-- BCP.EXE - You can specify dir
  <add key="BCPFileDir" value="c:\SQLAzureMW\BCPData"/>             <!-- Input / Output file location
  <add key="DelOldBCPFiles" value="true"/>                          <!-- Deletes old BCP files if exis
  <add key="BCPFileExt" value="dat"/>                               <!-- BCP output file extension -->
  <add key="DeleteBCPOutputFiles" value="false"/>                   <!-- If true, SQLAzureMW will dele
  <add key="TimeToSleepOnConnectionReset" value="2500"/>            <!-- Number of miliseconds to slee
  <add key="TimeBetweenChunks" value="15000"/>                      <!-- Number of miliseconds to slee
  <add key="ChunkSize" value="500000"/>                             <!-- Number of records to process
  <add key="BCPArgsIn" value="{0} in {1} -E -n -b 10000 -a 16384"/> <!-- BCP in command parameters -->
  <add key="BCPArgsOut" value="&amp;quot;{0}&amp;quot; out {1} -E -n"/>     <!-- BCP out command parameters --
  <add key="NotSupportedByAzureFileName" value="NotSupportedByAzureFile.Config"/>
  <add key="NumberOfBCPThreads" value="4"/>
  <add key="BCPSQLAzureErrorCodesRetry" value="SQLState\s=\s(40001|40174|40197|40501|40544|40549|40550|
```

Figure 2: App.config

3) **BCP Settings**
   a. **BCPFileExt** – Lets you specify the extension of your BCP output file.
   b. **BCPFileDir** – This specifies where you want BCP to put the data extracted from the source database.
   c. **DelOldBCPFiles** – This flag tells SQLAzureMW if it should write over any existing files in the BCPFileDir or create a new name. Set this value to "true" if you want old files overwritten or "false" if you want to keep the old files and just have SQLAzureMW create new ones.
   d. **DeleteBCPOutputFiles** – If you set this to "true", then SQLAzureMW will delete the BCP data file after it successfully uploads the data to the target server. If you want to do multiple uploads with the same script / data files, be sure this is set to false.
   e. **TimeToSleepOnConnectionReset** – SQLAzureMW has retry logic built into it. If for some reason the connection between SQLAzureMW and the target server is severed, SQLAzureMW will reset and try again. This value lets you specify the amount of time to wait between the time the connection was lost and the time that SQLAzureMW tries to connect again.
   f. **TimeBetweenChunks –** Specifies the time to sleep between BCP chunks when uploading to Azure SQL DB. The reason that you need this is to give Azure SQL DB time to process the previous chunk of data. If you do not give Azure SQL DB enough time to catch up, it will shut off your connection causing BCP to error out. This time is specified in milliseconds. The default value is 15,000 milliseconds (15 seconds).
   g. **ChunkSize –** This is the number of records that you want to process per BCP command. The default value is 100,000,000 records.
   h. **BCPArgsIn** – SQLAzureMW uses BCP to export your data from your local. SQLAzureMW uses {0} to specify Database.Owner.Table and it uses {1} to specify the output file. If you remove either one of these, then the export of data via BCP will fail. At runtime, SQLAzureMW will determine how you connect to your local database (trusted or via user credentials). It will append that information to BCPArgsIn. What you can do is change the –E –n –b to parameters better suited for you. For example, the –b switch specifies the batch size (number of records between commits). Do not confuse this with ChunkSize. For example, if I have a ChunkSize of 250,000 records, I might want BCP to do a commit after every 10,000 records. The 10,000 records per commit is your batch size.
      a. ***** NOTE ***** SQLAzureMW will add the –q (quoted identifier) if the database, schema, and table do not contain any periods (i.e [A.Database].[With.Schema].[My.Table]). You will notice that there are periods in the database name, schema name, and table name. There is a unique combination in that if you have periods in these names it could cause BCP to error out. It is highly recommended that you do not have periods in these names. The real problem is when you have a period in a table name and you have an index on a computed column. BCP cannot handle this combination.
   i. **BCPArgsOut** – SQLAzureMW uses BCP to upload data from your local database to Azure SQL DB. SQLAzureMW uses {0} to specify Database.Owner.Table and it uses {1} to specify the output file. If you remove either one of these, then the upload of data via BCP will fail. As with BCPArgsIn, you can change the –E –n to parameters better suited for you.
      a. ***** NOTE ***** Do not use –q (quoted identifier)
   j. **NotSupportedByAzureFileName** – Specifies the name of the NotSupportedByAzureFileName
   k. **NumberOfBCPThreads** – Specifies the number of parallel BCP upload process. If you specify 1, then the data upload will be done in a serial process. If you specify 4, the first 4 BCP commands in the script will be executed while the other BCP commands will be queued.
4) SQLAzureMW uses the output of BCP to look for errors and determine the number of records uploaded. The BCP Output in english looks like this in english:

```
Starting copy...
1000 rows successfully bulk-copied to host-file. Total received: 1000
1000 rows successfully bulk-copied to host-file. Total received: 2000
2155 rows copied.
```

Network packet size (bytes): 4096
Clock Time (ms.) Total : 1 Average : (2155000.00 rows per sec.)

The error output looks something like this in english:

5000 rows sent to SQL Server. Total sent: 150000
5000 rows sent to SQL Server. Total sent: 155000
SQLState = S1000, NativeError = 21
Error = [Microsoft][SQL Server Native Client 10.0][SQL Server]Warning: Fatal error 40501 occurred at Oct 30 2009 4:15PM. Note the error and time, and contact your system administrator.
BCP copy in failed

If your system is running a language other than one of the following languages:

- Chinese (Simplified, PRC) - CN
- Chinese (Taiwan) - TW
- Dutch – NL
- Dutch (Belgim) – BE
- English – EN
- French – FR
- German – DE
- Italian – IT
- Japanese – JP
- Portuguese (Brazil) – BR
- Portuguese (Portugal) – PT
- Russian – RU
- Spanish – ES

you will need to modify BCPSQLState, BCPError, BCPTotalSent and BCPSummary to reflect your language. For example if your system is set to French, then you would change BCPRowsCopied to "[0-9]+\slignes copiées". The best way to figure out what you need is to do a simple BCP in and out and capture the output string like above and then replace the Regex Search with your language specific output. Here is an example of what it looks like in the config file:

```
    af-ZA - Afrikaans (South Africa)
    de-DE - German (Germany)
    es-ES - Spanish (Spain)
    fr-FR - French
    it-IT - Italian
    ja-JP - Japanese
    nl-NL - Dutch (Dutch)
    nl-BE - Dutch (Belgium)
    pt-BR - Portuguese (Brazil)
    pt-PT - Portuguese (Portugal)
    ru-RU - Russian
    zh-CN - Chinese (Simplified, PRC)
    zh-TW - Chinese (Taiwan)
  -->
<af-ZA>...</af-ZA>
<de-DE>
  <add key="BCPRowsCopied" value="[0-9]+\sZeilen kopiert"/>                <!-- Regex Search -->
  <add key="BCPWarning" value="Warning:"/>                                 <!-- Regex Search -->
  <add key="BCPError" value="Error ="/>                                    <!-- Regex Search -->
  <add key="BCPSQLState" value="SQLState ="/>                              <!-- Regex Search -->
  <add key="BCPTotalSent" value="Insgesamt gesendet: [0-9]+\W+SQLState ="/> <!-- Regex Search -->
  <add key="BCPNumber" value="[0-9]+"/>                                    <!-- Regex Search -->
  <add key="BCPSummary" value="Zeit[\w\W]+sec.\)"/>                        <!-- Regex Search -->
  <add key="BCPCodePage" value="437"/>
</de-DE>
<en-US>...</en-US>
<es-ES>...</es-ES>
<fr-FR>...</fr-FR>
<it-IT>...</it-IT>
<ja-JP>
  <add key="BCPRowsCopied" value="[0-9]+\s行コピーされました。"/>     <!-- Regex Search -->
  <add key="BCPWarning" value="Warning:"/>                           <!-- Regex Search -->
  <add key="BCPError" value="Error ="/>                              <!-- Regex Search -->
  <add key="BCPSQLState" value="SQLState ="/>                        <!-- Regex Search -->
  <add key="BCPTotalSent" value="Total sent: [0-9]+\W+SQLState ="/>  <!-- Regex Search -->
  <add key="BCPNumber" value="[0-9]+"/>                              <!-- Regex Search -->
  <add key="BCPSummary" value="Clock[\w\W]+sec.\)"/>                 <!-- Regex Search -->
  <add key="BCPCodePage" value="Shift_JIS"/>
</ja-JP>
<nl-NL>...</nl-NL>
```

Figure 3: BCP Language Translation

5) **Options** – The option sections allow you to set option values so that you don't have to set them over and over again if you run against multiple databases. There are two key options that you should look for:
   a. `<add key="ScriptTableAndOrData" value="ScriptOptionsTableSchemaData" />` – This options lets you specify one of the three following values:
      1. `ScriptOptionsTableSchema (Table Schema Only)`
      2. `ScriptOptionsTableSchemaData (Table Schema with Data)`
      3. `ScriptOptionsTableData (Data Only)`

      If you only want to migrate your database schema and not the data, be sure to set this select "ScriptOptionsTableSchema".
   b. `<add key="CompatibilityChecks" value="ScriptOptionsOverride"/>` – This options is an override option. Basically, it tells SQLAzureMW to do one of three things:
      1. **ScriptOptionsOverrideOverride:** *Perform All Compatibility Checks* – This tells SQLAzureMW to perform all of the compatibility checks regardless of what the individual settings are.
      2. **ScriptOptionsTableSchemaData:** *Use Default Settings* – This tells SQLAzureMW to use the individual settings as specified in the config file (which can be modified at runtime as well).
      3. **ScriptOptionsTableData:** *Override: Do Not Perform Compatibility Checks* – This tells SQLAzureMW to skip all compatibility checks. **Note do not select this option unless your target server is SQL Server.**

      You might want to set this to "**ScriptOptionsTableSchemaData**" and only select the things TSQL Check options that you know (or think) might apply to your database. For example, one of the TSQL Check groups is Replication. If you don't use replication in your database, then there would be no need to spend CPU cycles looking for something that would not exist. To see what is being searched for in each group, look at the NotSupportedByAzureFile.config.

Running SQLAzureMW

When running SQLAzureMW, you will be asked to select a process. You can choose between four processes:

1) Analyze Only
   a. SQL Server – SQL Server instance in which you have the database that you wish to analyze.
   b. TSQL file – This is a text file that contains valid TSQL (DDL / DML) that you want to analyze for compatibility issues. This file can be generated by SQL Server Management Studio, SQLAzureMW, or any other application (or by hand) that generates valid DDL / DML.
   c. SQL Profiler trace file – Sometimes, analyzing the database for compatibility is not enough. There are a lot of programs that have dynamic SQL in the code. You can use SQL Profiler to capture as many different TSQL commands as possible (not different, not the same command 10,000 times over) and then use SQLAzureMW to analyze the trace file for compatibility issues.
2) Analyze and Migrate
   a. SQL Server – SQL Server instance in which you have the database that you wish to analyze and migrate to Azure SQL DB.
   b. TSQL file – This is a text file that contains valid TSQL (DDL / DML) that you want to analyze for compatibility issues. This file can be generated by SQL Server Management Studio, SQLAzureMW, or any other application (or by hand) that generates valid DDL / DML.
3) Migrate without Analyzing
   a. SQL Server – SQL Server instance in which you have the database that you wish to migrate to SQL Server. *** Note *** this option is really intended for SQL Server / Azure SQL DB to SQL Server migration. .
4) Run TSQL without Analyzing
   a. TSQL file – This is a text file that contains valid TSQL (DDL / DML) that you want to run against Azure SQL DB without analyzing. This file can be generated by SQL Server Management Studio, SQLAzureMW, or any other application (or by hand) that generates valid DDL / DML.

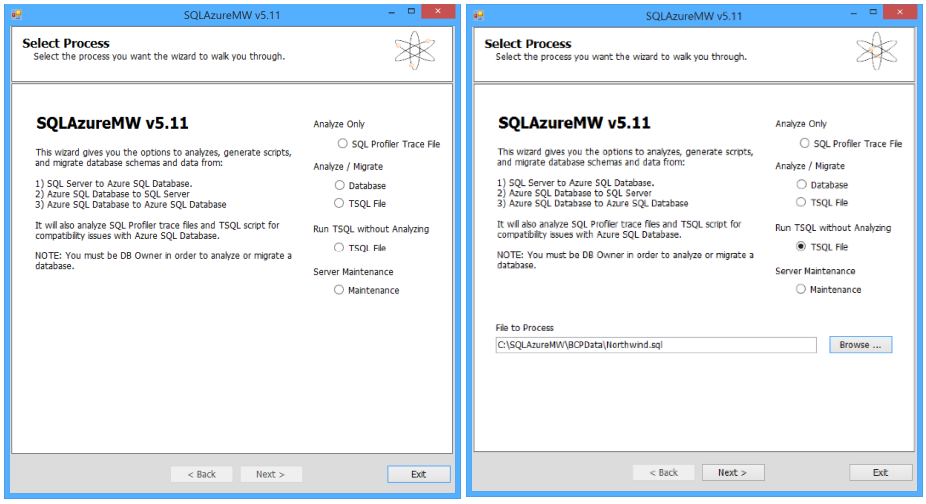

Figure 4: Select Process

Figure 5, shows using SQL Server as the source. You will be prompted to enter the server name and specify if you want to use Windows Authentication or supply your own SQL Server user credentials.

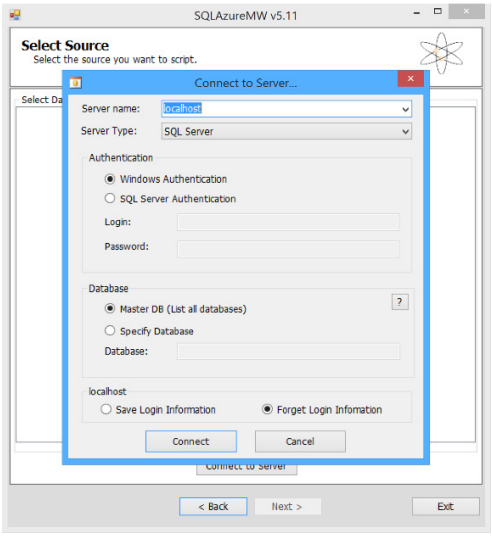

Figure 5: Connect to SQL Server

In order to script a database, you will have to have dbowner rights. If you connect to the master database, all databases will be listed on that specific instance and the databases you don't own will be grayed out, while the databases you do own will be black and allow you to select them. If you want, you can connect to a specific database and only that database will be displayed. After you tell SQLAzureMW to connect to your SQL Server instance, SQLAzureMW will list the databases attached to that instance (see Figure 6). You can then double click on the database (or select database and select "Next >").

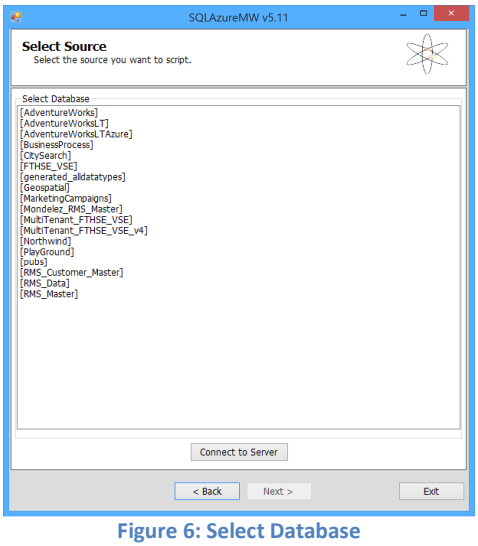

Figure 6: Select Database

Once you have selected your database, all of the objects available for scripting will be displayed. By default, all of the objects are selected for scripting. If you want, you can choose "Select specific database objects" and manually choose which objects you want to analyze / migrate. Note that any database object that is encrypted cannot be analyzed or migrated thus it is highlighted in yellow with red text to bring this to your attention.

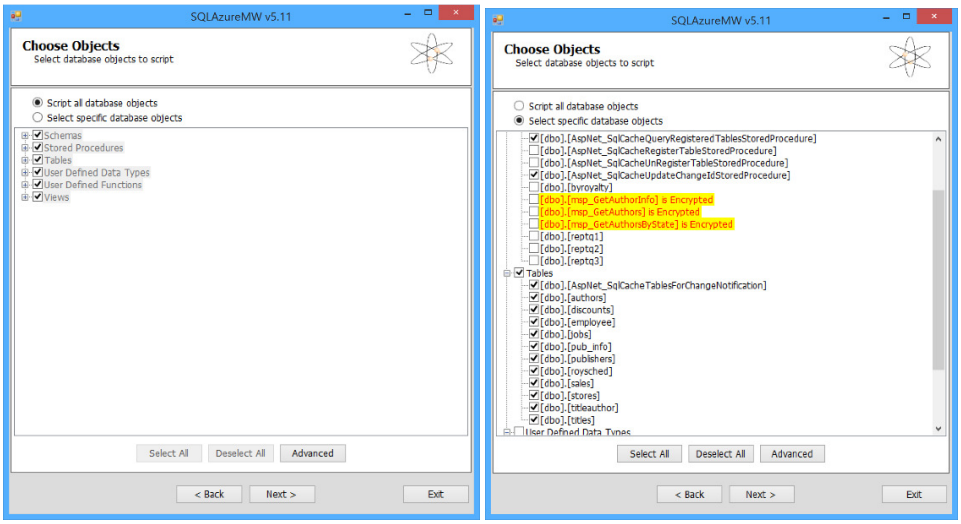

Figure 7: Choose Objects

You will notice that this menu has an “Advanced” button. Select this option if you want to modify any of the scripting options (like scripting your data). A dialog box will be displayed giving a list of scripting options (see Figure 8). You can modify database scripting options by selecting from the available settings in the list box to the right of each option. The options are applied globally to the objects that are being scripted.

Advanced Settings

General

Include If NOT EXISTS

When True, script includes a statement to check whether the object already exists in the database, and does not try to create a new object if the object already exists. Default is True. *** NOTE *** This feature is not supported by SQL Server 2014 and Azure SQL DB. If you need to have SQLAzureMW generate the IF NOT EXISTS statements, then you will need to use SQLAzureMW v3.x or SQLAzureMW v4.x.

Script Defaults

Includes default values for columns, when present in the original object. Default is True. For more information, see CREATE DEFAULT (Transact-SQL).

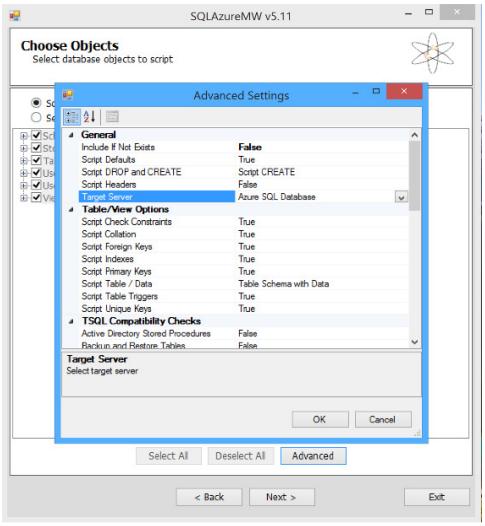

Figure 8: Scripting Options

Script DROP and CREATE

Default for this setting "Script CREATE" which is used when migrating to a new database. If you want to migrate to an existing database, you can have SQLAzureMW generate drop statements so that if the objects exists, then they would be dropped and then regenerated. *** NOTE *** that while this works, it is recommended that you only do this for all tables. The problem here is that if you select only specific tables to drop, you could have some dependency issues and not all of the dependent tables would be drop thus causing SQL Server / Azure to kick out errors. So be very careful when using anything but the default.

Script Headers

When True, descriptive comments are added to the script separating the script into sections for each object. Default is False.

Target Server

Use "Target Server" to specify your target server type. This tells SQLAzureMW which rules to check against. There are two types of target servers:

1) Azure SQL Database
2) SQL Server

Table/View Options

The following options apply only to scripts for tables or views.

Script Check Constraints

Adds CHECK constraints to the script. Default is True. CHECK constraints require data entered into a table to meet some specified condition. For more information, see CHECK Constraints.

Script Table / Data

Allows the user to specify:

1) Script table schema only
2) Script table schema and data
3) Export data only

This option will only be applied to the selected tables. If data is being exported, the data will be exported into a file and a comment will be inserted into the generated SQL that flags the output file name. Note, that this comment "`-- BCPArgs`" is required by SQLAzureMW to notify it what it needs to upload to Azure SQL DB. If you delete this comment or modify it, then the upload process will not be executed or fail.

Script Foreign Keys

Adds foreign keys to the script. Default is True. Foreign keys indicate and enforce relationships between tables. For more information, see FOREIGN KEY Constraints.

Script Indexes

Scripts the creation of indexes on the tables. Default is True. Indexes help you find data quickly.

Script Primary Keys

Scripts the creation of primary keys on the tables. Default is True. Primary keys uniquely identify each row of a table. For more information, see PRIMARY KEY Constraints.

Script Unique Keys

Scripts the creation of unique keys on the tables. Unique keys prevent duplicate data from being entered. Default is True. For more information, see UNIQUE Constraints.

TSQL Compatibility Checks

These section list 30 categories of compatibility checks to perform. I.E:

- Active Directory Stored Procedures
- Backup and Restore Tables
- Change Data Capture
- Log Shipping
- Replication

There is one special flag in this section that should be noted and that is the "Compatibility Checks". This options is an override option. Basically, it tells SQLAzureMW to do one of three things:

1) **Override: Perform All Compatibility Checks** – This tells SQLAzureMW to perform all of the compatibility checks regardless of what the individual settings are.
2) **Use Default Settings –** This tells SQLAzureMW to use the individual settings as specified in the config file (which can be modified at runtime as well).
3) **Override: Do Not Perform Compatibility Checks –** This tells SQLAzureMW to skip all compatibility checks.

Once you have selected the database objects to script and you are satisfied with the advanced settings, hit next to continue. SQLAzureMW display a summary of database objects and settings. When ready hit next. You will be prompted to verify that you want to proceed. Just select yes to continue.

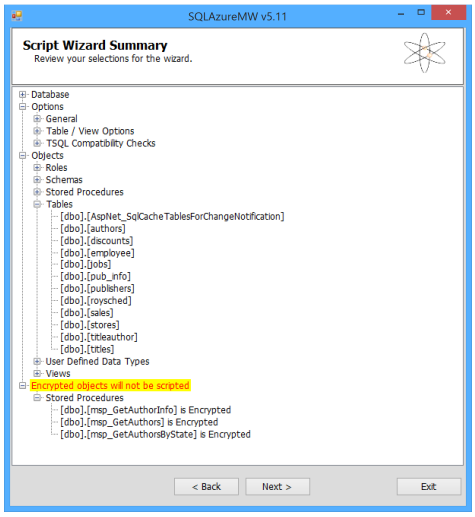

Figure 9: Wizard Summary

SQLAzureMW will then generate the script for the selected objects. As the TSQL script is generated for each object, SQLAzureMW will apply the Regex patterns defined in NotSupportedBySQLAzureFile.config. It will take the actions as define in the config file and then display the results in the Results Summary. The Results Summary window will have two tabs. The first tab, "Result Summary" shows the messages as defined in the NotSupportedBySQLAzureFile.config as well as the results from the BCP output (if you chose to export the data). Be sure to look through the Result Summary tab and look for anything in Red. You must resolve the red issues before running against Azure SQL DB.

***** NOTE *****

As of the writing of this paper, Azure SQL DB requires that a table has a clustered index. SQLAzureMW will evaluate each table scripted for a clustered index and if it does not find one, it will generate a clustered index for you (see Figure 10). It will output a message in the "Result Summary" tab and tell you that a clustered index was generated for you and suggest that you look at the generated clustered index and make sure that it works for you.

The generated TSQL will be displayed in the "SQL Script" tab. Along with the generated TSQL, SQLAzureMW will display commented messages starting with "--~". This sequence is a tag used by SQLAzureMW to identify comments to be removed before running against Azure SQL DB. These comments are for your information so that you can see where automatic changes occurred against your objects without having to switch back to the Result Summary tab and then trying to find the exact place in the SQL Script tab. The "-- BCPArgs" is also a special tag that tells the upload process where and at what point BCP should be run to upload data. If you remove this tag or modify it, then the data will not be uploaded.

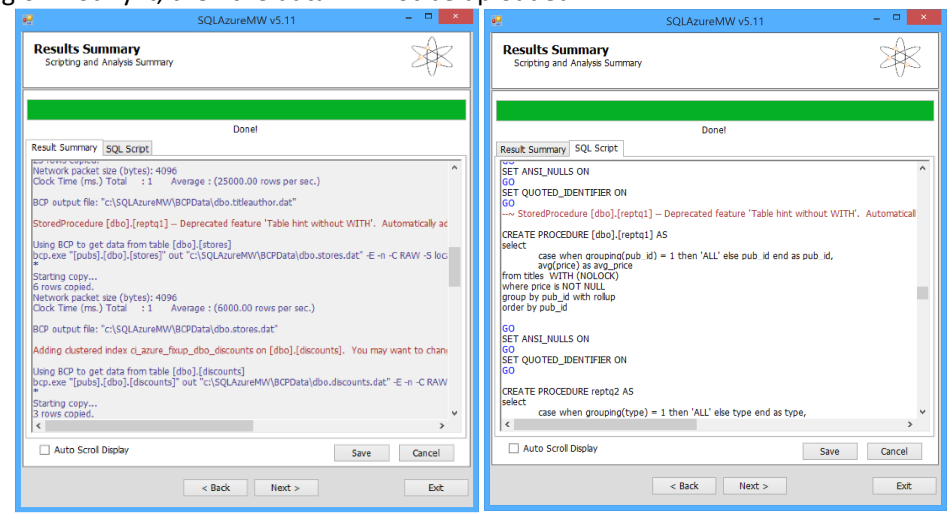

Figure 10: Results Summary

Once you have resolved all the issues (if there is any), you can have SQLAzureMW run the TSQL script against Azure SQL DB by selecting “Next >”. Also note, that you can save the results summary and SQL script to a file for offline work or history. You can also save the script at this point for use by SQLAzureMWBatch. A good example of why you would want to do this is if you have a Test / QA environment where you want to test, drop database, and reload. To see more on this topic, see SQLAzureMWBatch.

Enter your target server connection information. Note that the information will be entered for you if you specified it in the App.config file (see the App.config section above).

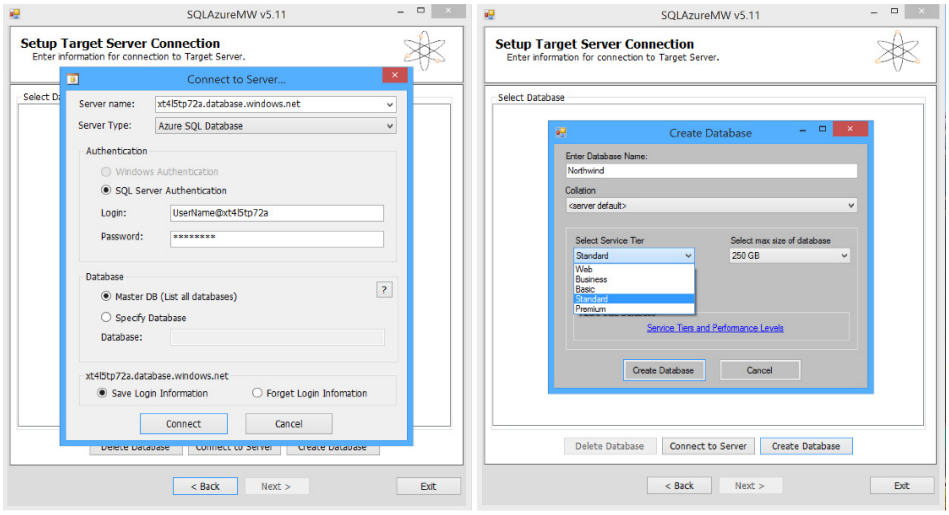

Figure 11: Target Server Connection

After your target connection information has been entered, hit the “Connect” button. This will verify that your connection information is correct and list any databases that you have in your target server. You can then select an existing database to have the TSQL run against, delete an old database and / or you can create a new database. Note that when you “Select Service Tier”, you will have the option to select Web, Business, Basic, Standard, and Premium service tiers. Web and Business editions should not be used for production databases since both Web and Business editions have been retired and will be removed by Sept 2015. After you have selected service tier, performance edition, collation, database size, and entered your new database name, hit “Create” to create your database. You should now see your new database on the screen. Select your new database and when you are ready, select the “Next >” button. You will be prompted to verify that you want to run the script that was just generated against your selected database.

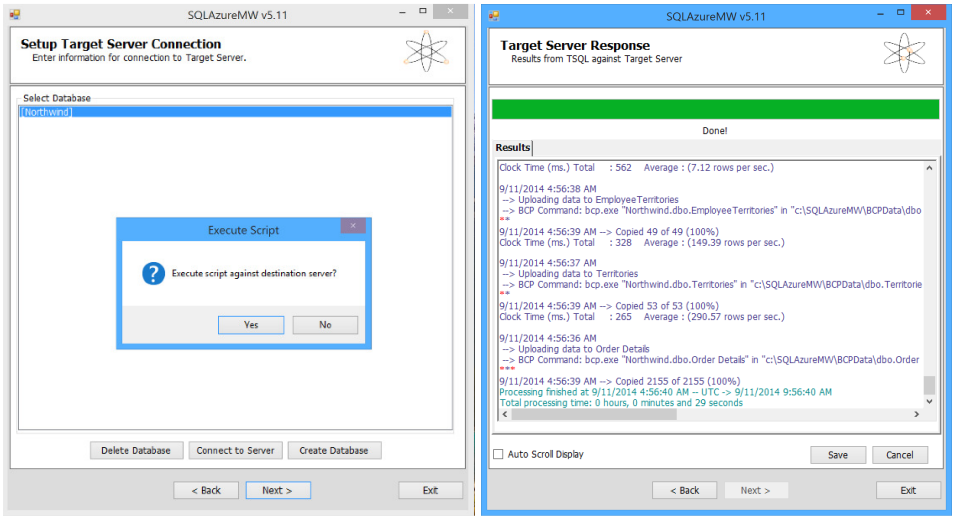

Figure 12: Target Server Response

SQLAzureMW will then execute the generated TSQL script against target server and display the results in the "Target Server Response" window. Note that if there is an error reported back from the target server, the error message will be displayed in red along with the TSQL that caused the error so that you can evaluate it and take action to resolve. If there are no red messages displayed, then your database has been successfully migrated to the target server and you can exit the program.

Select Source – TSQL File

You can use this option to have SQLAzureMW load a text file that contains valid TSQL (DDL / DML). This file can be generated by SQL Server Management Studio, SQLAzureMW, or any other application (or by hand) that generates valid DDL / DML. You will be asked to enter the directory / filename of the input file.

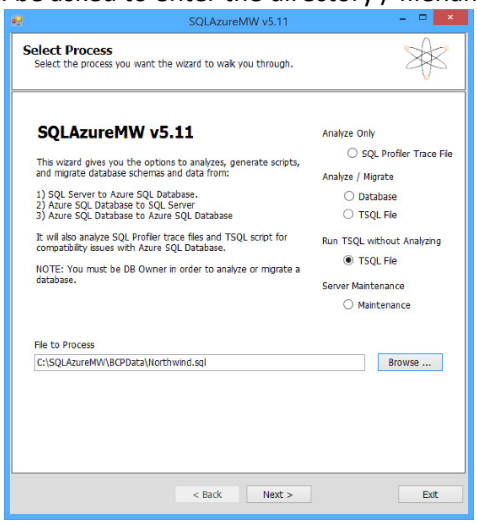

Figure 13: Select Process – TSQL File

When selecting a TSQL File, you can choose between three processes:

1) **Analyze Only** – This is a text file that contains valid TSQL (DDL / DML) that you want to analyze for compatibility issues. This file can be generated by SQL Server Management Studio, SQLAzureMW, or any other application (or by hand) that generates valid DDL / DML.
2) **Analyze and Migrate** – This is a text file that contains valid TSQL (DDL / DML) that you want to analyze for compatibility issues. This file can be generated by SQL Server Management Studio, SQLAzureMW, or any other application (or by hand) that generates valid DDL / DML.
3) **Run TSQL without Analyzing** – This is a text file that contains valid TSQL (DDL / DML) that you want to run against Azure SQL DB without analyzing. This file can be generated by SQL Server Management Studio, SQLAzureMW, or any other application (or by hand) that generates valid DDL / DML.

Note that the Advanced button will not be available for you to modify the check compatibility options. If you want to modify these options, then edit the SQLAzureMW.exe.config before you run SQLAzureMW.

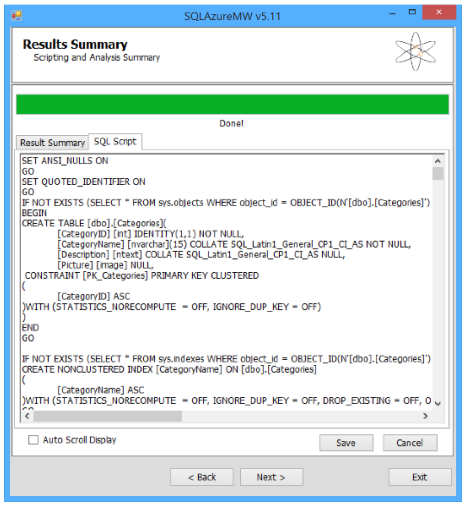

Figure 14: Results Summary

SQLAzureMW will analyze or not analyze (depending on your selected process) and display the analysis results (if any) in the "Results Summary" tab and the finished TSQL script in the "SQL Script" tab.

Look at the "Result Summary" tab and if you see any issues you can fix before you proceed. Once you have resolved all the issues (if there is any), you can have SQLAzureMW run the TSQL script against the target server by selecting "Next >".

Enter your target server connection information. Note that the information will be entered for you if you specified it in the App.config file (see the App.config section above).

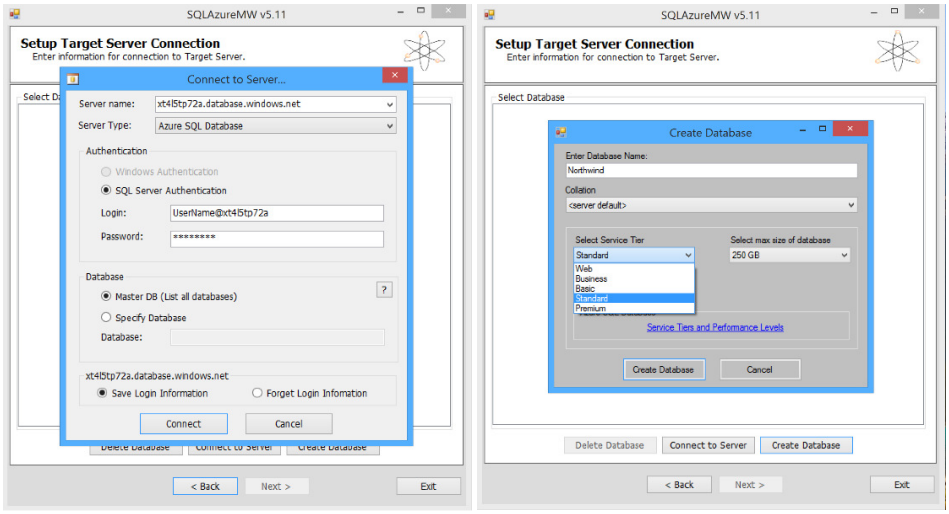

Figure 15: Azure SQL DB Connection

After your target connection information has been entered, hit the "Connect" button. This will verify that your connection information is correct and list any databases that you have in your target server. You can then select an existing database to have the TSQL run against, delete an old database and / or you can create a new database. Note that when you "Select Service Tier", you will have the option to select Web, Business, Basic, Standard, and Premium service tiers. Web and Business editions should not be used for production databases since both Web and Business editions have been retired and will be removed by Sept 2015. After you have selected service tier, performance edition, collation, database size, and entered your new database name, hit "Create" to create your database. You should now see your new database on the screen. Select your new database and when you are ready, select the "Next >" button. You will be prompted to verify that you want to run the script that was just generated against your selected database.

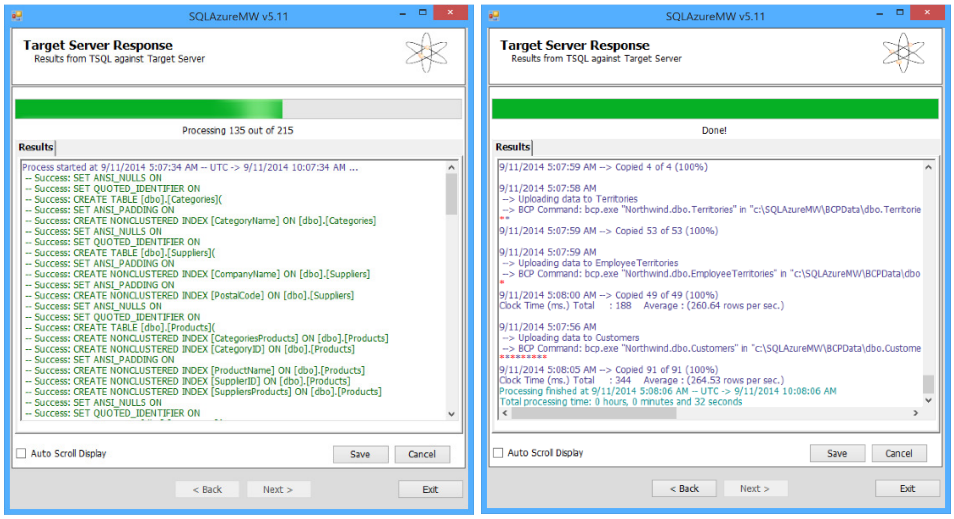

Figure 16: Target Server Response

SQLAzureMW will then execute the TSQL script against the target server and display the results in the "Target Server Response" window. Note that if there is an error reported back from the target server, the error message will be displayed in red along with the TSQL that caused the error so that you can evaluate it and take action to resolve. If there are no red messages displayed, then your database has been successfully migrated to the target server and you can exit the program.

*** NOTE *** SQLAzureMW will display the actual BCP command. So, if you do run into an error, you can copy the BCP command and run it from a command prompt.

Select Source – SQL Profiler Trace File

You can use this option to have SQLAzureMW analyze a trace file created by SQL Profiler. You will be asked to enter the directory / filename of the input file.

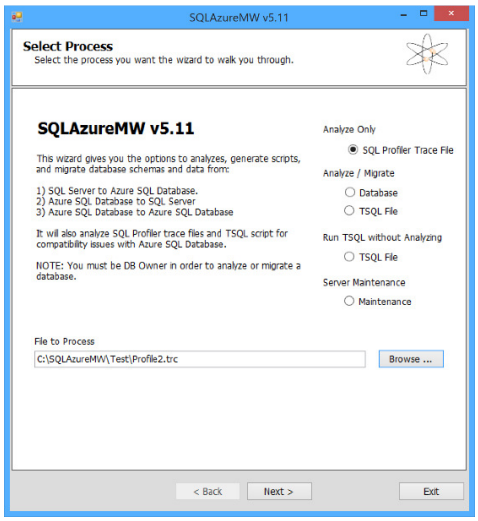

Figure 17: Select Process – Trace File

The only process available to the trace file is "Analyze Only".

Note that the Advanced button will not be available for you to modify the check compatibility options. If you want to modify these options, then edit the SQLAzureMW.exe.config before you run SQLAzureMW.

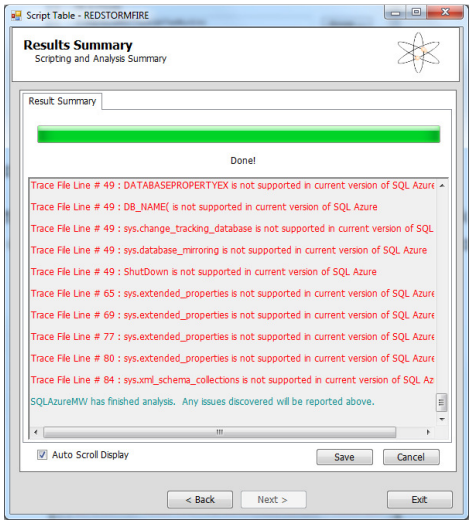

Figure 18: Results Summary

As can be seen above, when SQLAzureMW encounters a compatibility issue, it flags the error (i.e. sys.database_mirroring) and tells you what line it can be found in the trace file. You can edit the trace file, hit Ctrl G and then enter the line number you want to go to. From there, you can see the whole TSQL statement that was flagged as an issue. From there, you can make the changes to your database to make it compatible with your target server.