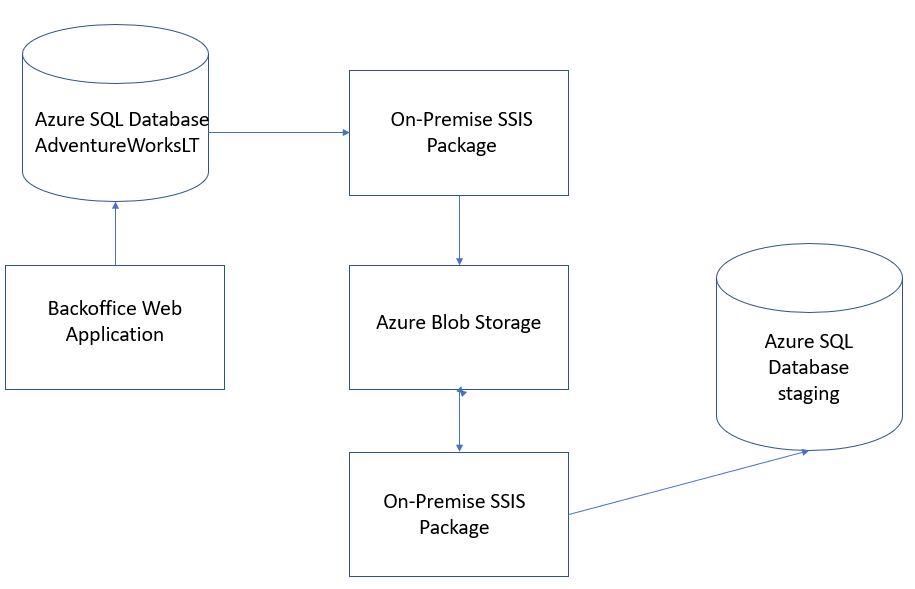
# Using Azure Storage and Azure SQL Database for SSIS Development

##### Problem

I am starting a new data warehousing project for a new client where I will use SQL Server Integration Services (SSIS) for the Extract, Transform and Load (ETL) operations. The client wants to create a hybrid environment for development where I will only install SQL Server Data Tools (SSDT) and SQL Server Management Studio (SSMS) on my laptop. The goal is to leverage Azure for file storage and the SQL Server databases. Since this is a development environment, I will use Azure SQL Database for the databases. Ultimately my client wants to transition this ETL application to [Azure Data Factory](https://azure.microsoft.com/en-us/services/data-factory/) at some point in the future. Can you walk me through the steps to accomplish this?

##### Solution

The following diagram shows the solution to be built in this tip:



The main points are:

* An On-Premise SSIS package extracts data from the Backoffice Web Application’s Azure SQL Database and writes pipe-delimited files to Azure Blob Storage on a daily basis for the various types of transactions that are processed and of interest to the data warehouse
* An On-Premise SSIS package reads the pipe-delimited files from Azure Blob Storage and inserts the records into staging tables in the Azure SQL Database

Note that for demonstration purposes, I’m creating two SSIS packages to walk through the scenario where the Backoffice Web Application and the ETL process are loosely integrated. Certainly, I could create a single SSIS package that extracts data from the Backoffice Web Application’s database and inserts that data into the staging database. I want to demonstrate extracting data from an Azure SQL Database and writing to Azure Blob Storage as well as reading data from Azure Blob Storage and inserting into an Azure SQL Database.

## Prerequisites

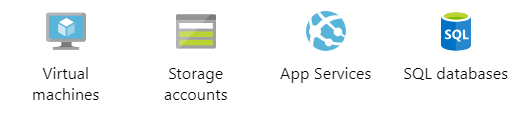
There are a couple of things that we need to take care of before we walk through building the on-premise SSIS packages:

* Create Azure Subscription
* Create Azure Storage Account
* Create Azure SQL Database
* Install SQL Server Data Tools (SSDT)
* Install Azure Feature Pack for SQL Server Integration Services (SSIS)
* Install SQL Server Management Studio (SSMS)

I will provide a brief description of the setup steps for each of these assuming that you are starting from scratch.

#### Create Azure Subscription

Go to <https://azure.microsoft.com/> and setup an account. You can probably get a free subscription for a period of time or sign up for a pay-as-you-go plan. After you have an Azure Subscription, go to <https://portal.azure.com/> and sign in. You will see the following on the home page (just the relevant portion of the page is shown):



#### Create Azure Storage Account

Click on the Storage accounts icon on the Azure Home Page shown above to create an Azure Storage Account. Follow the steps in [Create a storage account](https://docs.microsoft.com/en-us/azure/storage/common/storage-quickstart-create-account?tabs=azure-portal) to get your storage account created. Make a note of the Storage account name and Container name; you will need them later.

#### Create Azure SQL Database

Click on the SQL databases icon on the Azure Home Page shown above to create an Azure SQL Database. Follow the steps in [Quickstart: Create a single database in Azure SQL Database using the Azure portal](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-single-database-get-started) to get your Azure SQL Database created. Make a note of the Server name, Database name, Server admin name and password when you create a new server and database; you will need them later.

After you create your Azure SQL Database, make sure to [Create a server-level firewall rule](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-server-level-firewall-rule) so you can access the database outside of the Azure portal.

Note that when you create a new Azure SQL Database, you have the following options for the Source:

* Blank database
* AdventureWorksLT
* Backup

For this tip we need two databases: chose AdventureWorksLT for the application database (choose AdventureWorksLT as the Source) and a Blank database for the staging database.

#### Install SQL Server Data Tools (SSDT)

SSDT is a free, stand-alone version of Visual Studio that you can use to develop Analysis Services, SQL Server Integration Services and SQL Server Reporting Services project. Follow this link to [download and install SQL Server Data Tools (SSDT) for Visual Studio](https://docs.microsoft.com/en-us/sql/ssdt/download-sql-server-data-tools-ssdt?view=sql-server-2017).

#### Install Azure Feature Pack for SSIS

Download and install the [Azure Feature Pack for SSIS](https://docs.microsoft.com/en-us/sql/integration-services/azure-feature-pack-for-integration-services-ssis?view=sql-server-2017) which includes a handful of components for working with Azure from SSIS.

#### Install SQL Server Management Studio (SSMS)

SSMS is the tool we use to work with a SQL Server database. For this tip I am using version 14.0.17289.0. Follow this link to [download SQL Server Management Studio](https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-2017). I like to use the current general availability version. Note that if you have an older version of SSMS installed, it may not have the features for working with Azure SQL Databases.

## Get Azure Values

Before we start creating SSIS packages, we need to gather all of the parameter values from our Azure Storage Account and Azure SQL Databases that we will need for this tip. This section assumes these items have been created.

#### Azure Storage Account Values

Follow these steps to get the Azure Storage Account values we need:

1. Navigate to the [Azure home page](https://portal.azure.com/)
2. Click on Storage accounts
3. Select your storage account
4. Click on Access Keys (under Settings) and copy the key value under Key1
5. Click on Overview (top left)
6. Click on Blobs (under Services)
7. Select your Container name

The following are the parameter values for this tip based on the above steps (sensitive values shown as asterisks):

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| Storage Account Name | backofficestaging |
| Access Key | \*\*\*\*\*\*\*\*\*\*\* |
| Container Name | subscriptions |

#### Azure SQL Database Values

I use two Azure SQL Databases in this tip: an application database and a staging database. Follow these steps to get the Azure SQL Database parameter values that we need:

1. Navigate to the [Azure home page](https://portal.azure.com/)
2. Click on SQL databases
3. Select the database server and database name

The following are the parameter values for this tip based on the above steps (sensitive values shown as asterisks):

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| Application Database Server | \*\*\*\*\*\*\*\*\*\*\* |
| Application Database Name | AdventureWorksLT |
| User Name | \*\*\*\*\*\*\*\*\*\*\* |
| Password | \*\*\*\*\*\*\*\*\*\*\* |
| Staging Database Server | \*\*\*\*\*\*\*\*\*\*\* |
| Staging Database Name | staging |
| User Name | \*\*\*\*\*\*\*\*\*\*\* |
| Password | \*\*\*\*\*\*\*\*\*\*\* |

Note that the User Name and Password were specified when you created the server.

## Create SSIS Packages

Now we are ready to create the SSIS packages. Open SSDT and create a new SSIS project. Add new SSIS packages named EXTRACT\_SUBSCRIPTIONS and STAGE\_SUBSCRIPTIONS.

I will walk through the following steps to build the SSIS packages:

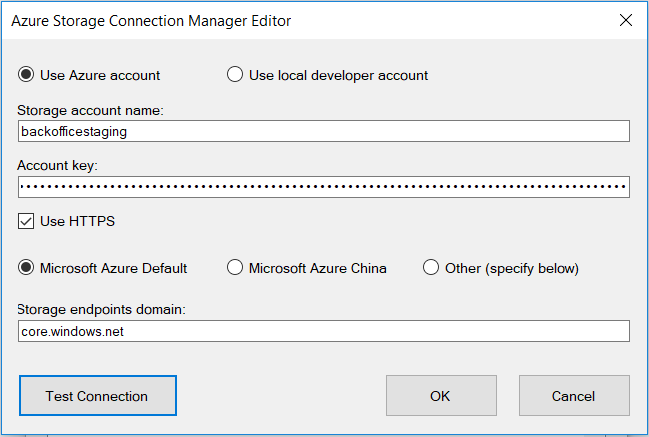
1. Create the Azure Storage Connection Manager
2. Create an OLE DB Connection Manager for an Azure SQL Database
3. Connect to Azure SQL Databases using SSMS
4. Create the SUBSCRIPTIONS table in the staging Azure SQL Database
5. Create the EXTRACT\_SUBSCRIPTIONS SSIS package
6. Create the STAGE\_SUBSCRIPTIONS SSIS package

#### Create the Azure Storage Connection Manager

We need to create an Azure Storage Connection Manager in order to save a delimited file to Azure Blob Storage and open a delimited file from Azure Blob Storage. Follow these steps to create the Azure Storage Connection Manager:

1. Click on the Control Flow tab in an SSIS package
2. Open an SSIS package or add one to your SSIS project
3. Right-click in the Connection Managers area of the SSIS package designer
4. Select New Connection from the popup menu
5. Select Azure Storage as the Connection Type
6. Click Add

The following form will be displayed (fill in the Storage account name and Account key from the Azure Storage Account Values section above):



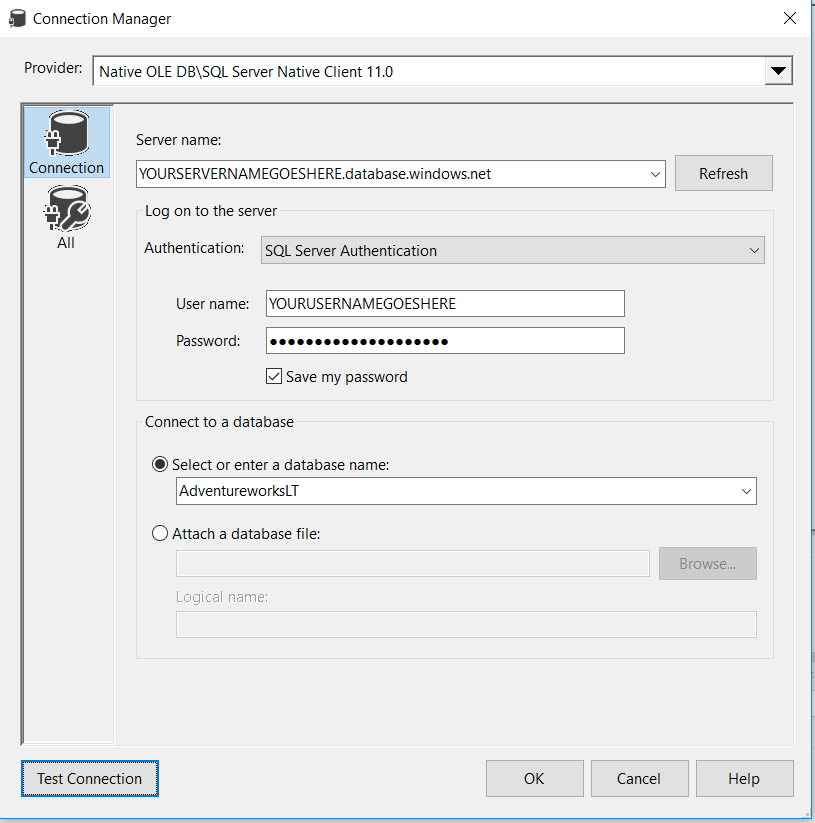
Click Test Connection to make sure you can connect to your Azure Storage Account. Click OK to save the Azure Storage Connection Manager. Finally right click on the connection manager and select Convert to Project Connection. This connection manager will be used in both of the SSIS packages we will create.

#### Create an OLE DB Connection Manager for an Azure SQL Database

There really isn’t much difference between creating an OLE DB connection manager for an Azure SQL Database and an OLEDB connection manager for an on-premise SQL Server database. Follow these steps to create an OLE DB connection manager for an Azure SQL Database:

1. Click on the Control Flow tab in an SSIS package
2. Right-click in the Connection Managers area of the SSIS package designer
3. Select New OLE DB Connection from the popup menu
4. Select New from the OLE DB Connection Manager form

The following form will be displayed (fill in the YOURXXX with your values from the Azure SQL Database Values section above):



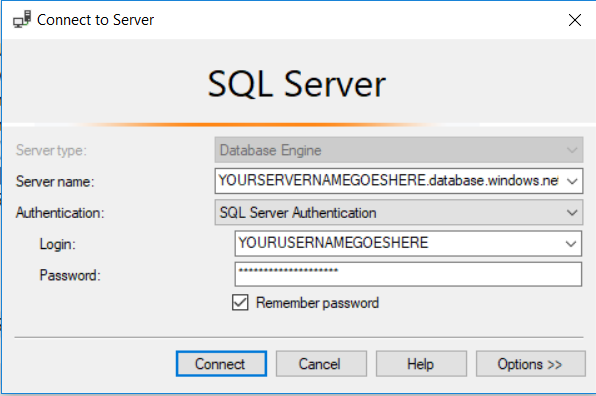
Note that the only difference for an Azure SQL Database connection versus an on-premise SQL Server database is that the server name has “.database.windows.net” appended to it.

Click Test Connection to verify that your connection manager works. Click OK to complete the Connection Manager form. Click OK again on the OLE DB Connection Manager form. Right-click on the Connection Manager and select Rename; change to AdventureWorksLT. Right-click on the Connection Manager and select Convert to Project Connection so that the connection manager will be available for use in any SSIS package in the project.

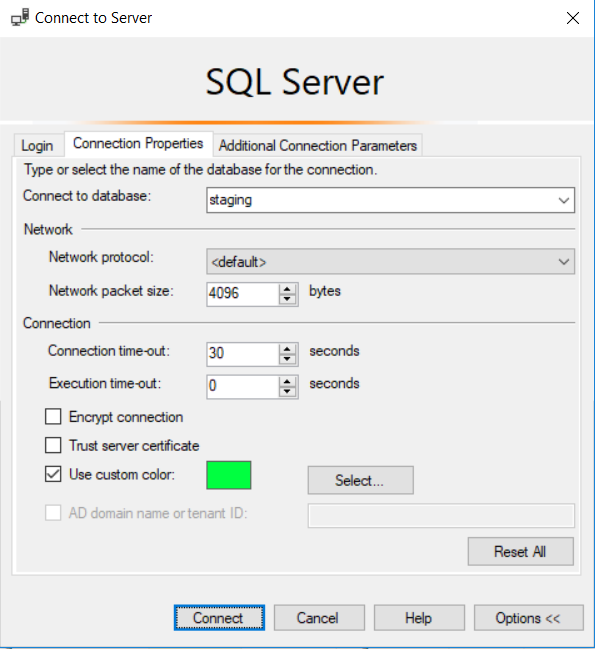
Repeat the above steps to create the staging connection manager.

#### Connect to Azure SQL Databases Using SSMS

Open SSMS, click Connect in the Object Explorer, and select Database Engine from the popup menu. Fill in the Connect to Server form replacing YOURXXX with the values from the Azure SQL Database Values section:



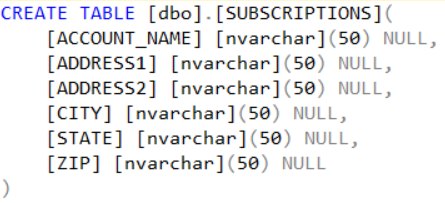
Click the Options button and fill in the Connect to database name as shown below:



Click Connect and you will see the staging database in the Object Explorer.

#### Create the SUBSCRIPTIONS table in the staging Azure SQL Database

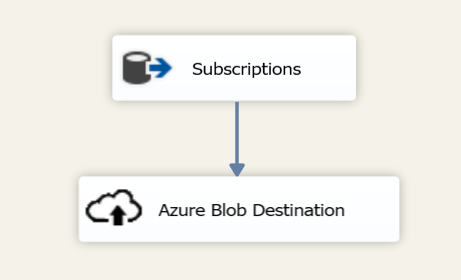
Connect to the staging Azure SQL Database using the steps in the previous section. Click on the staging database in the Object Explorer and click New Query on the toolbar; enter and execute the following script to create the SUBSCRIPTIONS table; you can copy and paste from the download:



#### Create the EXTRACT\_SUBSCRIPTIONS SSIS package

Open the EXTRACT\_SUBSCRIPTIONS package in your SSIS project (or add it if you haven’t already). This package is used to extract data from the Backoffice Web Application’s Azure SQL Database and write pipe-delimited files to Azure Blob Storage.

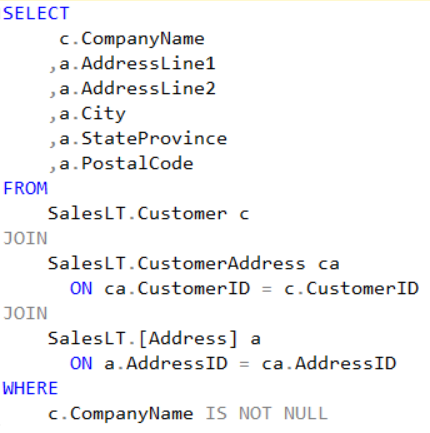
The Data Flow for the package is shown below:



The following are the main points on the above Data Flow:

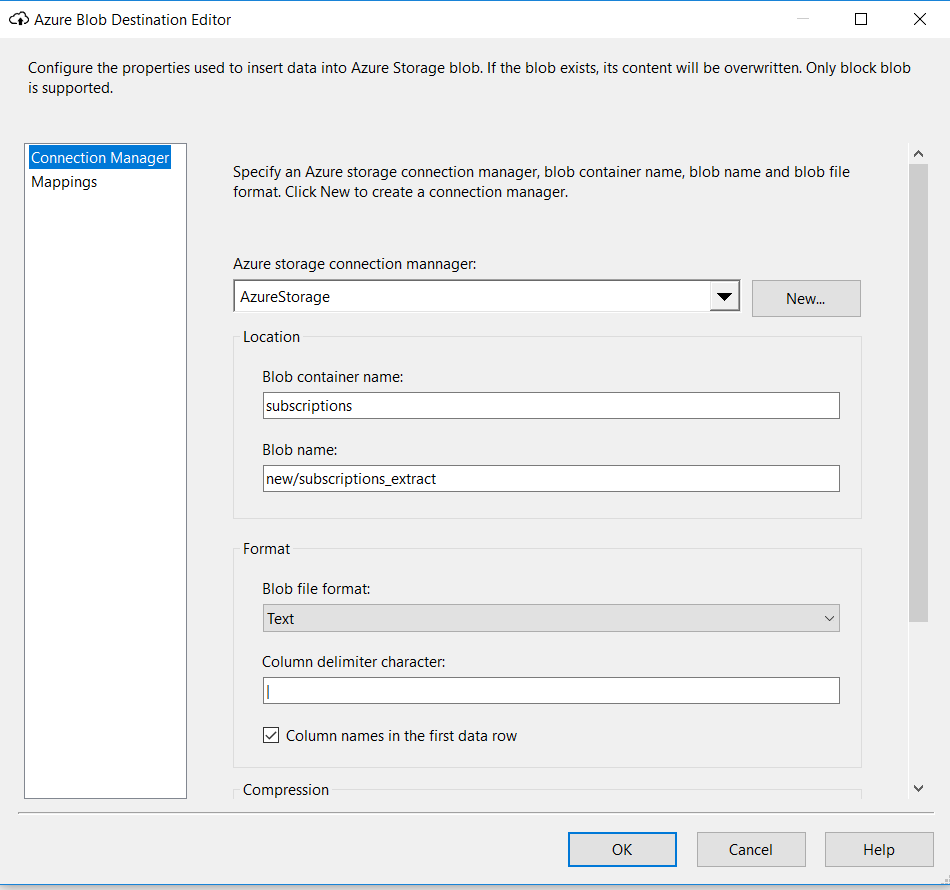
* Subscriptions is an OLE DB Source that executes the query below on the AdventureWorksLT Azure SQL Database
* The query results are written to Azure Blob storage using the Azure Blob Destination component which comes from the [Azure Feature Pack for SSIS](https://docs.microsoft.com/en-us/sql/integration-services/azure-feature-pack-for-integration-services-ssis?view=sql-server-2017)

The following query is used to extract data from the AdventureWorksLT Azure SQL Database (you can copy from the download code and paste into the OLE DB Source):



Since this is just a demonstration of using Azure Blob Storage in an ETL process, the above query does not have any filtering criteria to only grab the rows that have changed since the last time the query was run.

The following shows the configuration of the Azure Blob Destination component:

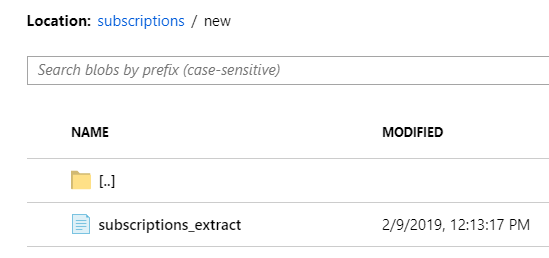


The Azure Blob Destination component is creating a pipe-delimited file that is saved to Azure Blob Storage. The following are the main points on the configuration of the Azure Blob Destination component:

* The Azure storage connection manager was configured above (Create the Azure Storage Connection Manager section)
* The Blob container name comes from the Azure Storage Account Values section
* The Blob name is what I used; it’s a folder name and a file name
* The file is text and will have column names in the first row

I found one limitation with the Azure Blob Destination component; it does not provide an Expressions property. Ideally, I might like the Blob container name and Blob name properties to be set at runtime via an expression. On the other hand, it’s a bit less complicated to always look for a specific file in a specific container.

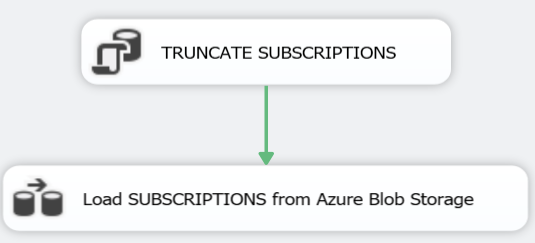
When you run this SSIS package, you can navigate to the [Azure home page](https://portal.azure.com/) and drill down into your storage account and see the file that was created in Azure Blob Storage; e.g.:



#### Create the STAGE\_SUBSCRIPTIONS SSIS package

Open the STAGE\_SUBSCRIPTIONS package in your SSIS project (or add it if you haven’t already). This package is used to read a pipe-delimited file from Azure Blob Storage and insert every record into the SUBSCRIPTIONS table in the staging Azure SQL Database.

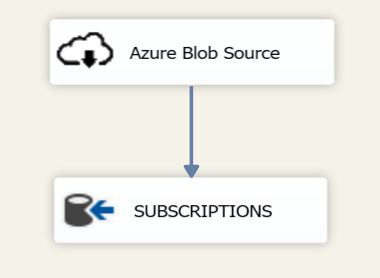
The following is the Control Flow for the package:



The following are the main points on the Control Flow:

* TRUNCATE SUBSCRIPTIONS is an Execute SQL task that truncates the SUBSCRIPTIONS table in the staging Azure SQL Database
* Load SUBSCRIPTIONS from Azure Blob Storage is a Data Flow detailed below

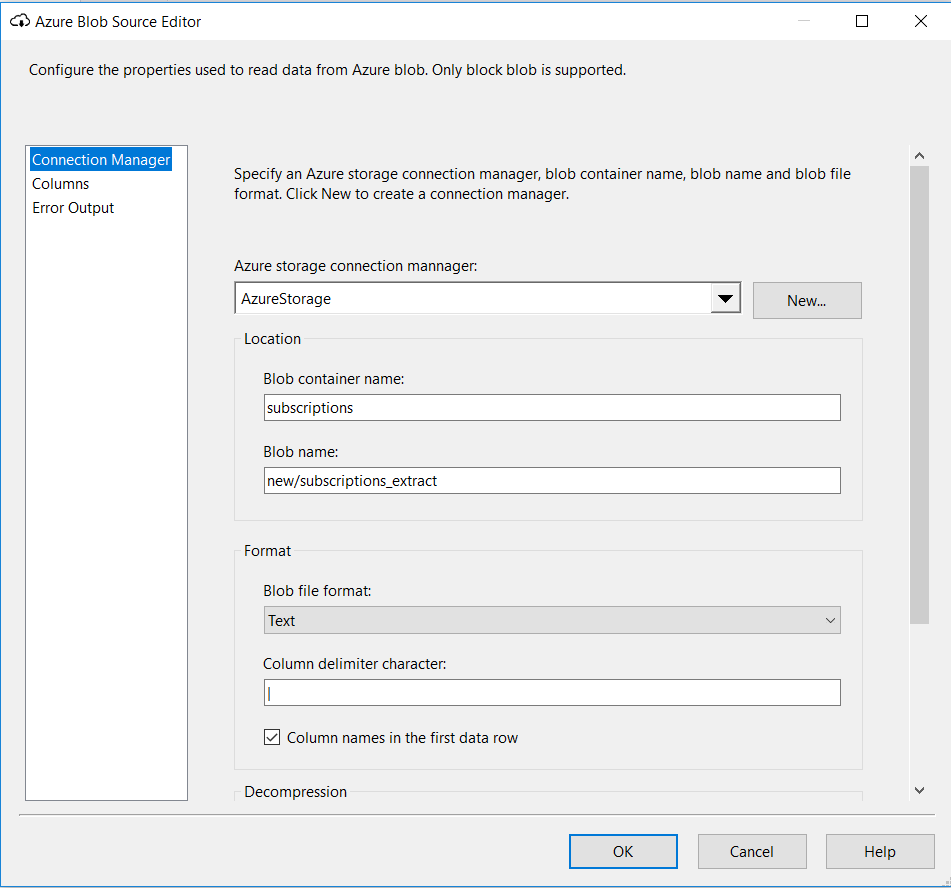
The following is the Load SUBSCRIPTIONS from Azure Blob Storage Data Flow:



The following are the main points on the above Data Flow:

* The Azure Blob Source component reads a pipe-delimited file from Azure Storage; it comes from the [Azure Feature Pack for SSIS](https://docs.microsoft.com/en-us/sql/integration-services/azure-feature-pack-for-integration-services-ssis?view=sql-server-2017)
* SUBSCRIPTIONS is an OLE DB Destination which inserts the rows in the pipeline into the SUBSCRIPTIONS table in the staging Azure SQL Database

The configuration of the Azure Blob Source component is shown below:



The following are the main points on the configuration of the Azure Blob Source component:

* The Azure storage connection manager was configured above (Create the Azure Storage Connection Manager section)
* The Blob container name comes from the Azure Storage Account Values section
* The Blob name is what I used; it’s a folder name and a file name
* The blob is a pipe-delimited text file with column names in the first row

The Azure Blob Source component has the same limitation as the Azure Blob Destination component; you can’t use Expressions to set any values at runtime.

After you run this SSIS package, you can query the SUBSCRIPTIONS table in the staging Azure SQL Database. In my case I see 417 rows.

## Conclusion

My goal for this tip was to demonstrate how you can leverage Azure Storage and Azure SQL Database in your development environment. As with any demo, there are always shortcuts taken; e.g. I didn’t address security that you would need in a production environment. However, I’m happy with the outcome and I plan to leverage Azure for my development efforts going forward.

While the sample code for this tip is available (see the link in the Next Steps section below), the SSIS project and packages have the Protection Level set to EncryptSensitiveWithUserKey. This is to keep the parameters of my Azure subscription private. When you open the project, you will get errors and warnings about not being able to decrypt things. If your try to run the SSIS packages, you will experience a significant delay before you get error messages that the package cannot execute due to errors. Your best bet is to create the packages from scratch.

##### Next Steps

* Download the sample code [TO DO: insert link] and experiment in your development environment.
* Take a look at the [Azure Tips Category](https://www.mssqltips.com/sql-server-tip-category/87/azure/) on [MSSQLTips.com](https://www.mssqltips.com/) to expand your knowledge in this area.