Using the .Net SDK to trigger data transformation

Overview

This article contains a walkthrough that you can follow to create a sample .NET console application that allows you to trigger a Data Transformation Job and track it for completion.

Prerequisites

- Visual Studio 2012 or 2013 or 2015
- Azure Powershell
- Configuration settings to initialize the Data Transformation Job (instructions to obtain them are in the walkthrough)
- A job definition which has been correctly configured in a Hybrid Data Resource within a Resource Group.
- All DLLs present in this folder on the github repository.
- Get-ConfigurationParams.ps1 script from the github repository.

Walkthrough

1. Steps to retrieve the configuration parameters –

    a. Download the Get-ConfigurationParams.ps1 from the github repository script in C:\DataTransformation
    b. Run the Get-ConfigurationParams.ps1 script from the github repository with the following command

    C:\DataTransformation\Get-ConfigurationParams.ps1 -SubscriptionName "AzureSubscriptionName" -ActiveDirectoryKey "anyrandompassword" -AppName "ApplicationName"

    [You can pass in any values for the ActiveDirectoryKey and the AppName]

    c. This script will output the following values –
        i. Client Id
        ii. Tenant Id
        iii. ActiveDirectoryKey (same as the one entered above)

iv. Subscription Id

2. Using Visual Studio 2012, 2013 or 2015, create a C# .NET console application.

   - Launch **Visual Studio 2012/2013/2015**.
   - Click **File**, point to **New**, and click **Project**.
   - Expand **Templates**, and select **Visual C#**.
   - Select **Console Application** from the list of project types on the right.
   - Enter **DataTransformationApp** for the **Name**.
   - Select **C:\DataTransformation** for the **Location**.
   - Click **OK** to create the project.

3. Now, add all DLLs present in the dlls folder as **References** in the project which you created just now. You will need to download the dll files.
   a. In Visual Studio, go to View -> Solution Explorer, click on the arrow to the left of DataTransformationApp project -> click References -> right click -> Add Reference -> Browse to the location of the packages folder, select all the DLLs then click Add, and then click Ok.

4. Add the following **using** statements to the source file (Program.cs) in the project.

```
using System;
using System.Collections.Generic;
using System.Threading;
using Microsoft.Azure.Management.HybridData.Models;
using Microsoft.Internal.Dms.DmsWebJob;
using Microsoft.Internal.Dms.DmsWebJob.Contracts;
```

5. The below code initializes the data transformation job instance, add this in the Main method, replace the values of configuration parameters as obtained earlier. Plug in the values of Resource Group Name and Hybrid Data Resource name. The Resource Group Name is the one which hosts the Hybrid Data Resource on which the Job Definition was configured.

```
// Setup the configuration parameters.
var configParams = new ConfigurationParams
{
        ClientId = "client-id",
        TenantId = "tenant-id",
        ActiveDirectoryKey = "active-directory-key",
        SubscriptionId = "subscription-id",
        ResourceGroupName = "resource-group-name",
        ResourceName = "resource-name"
};

// Initialize the Data Transformation Job instance.
DataTransformationJob dataTransformationJob = new DataTransformationJob(configParams);
```

6. Specify the parameters with which the job definition needs to be run

```
string jobDefinitionName = "job-definition-name";


DataTransformationInput dataTransformationInput =
dataTransformationJob.GetJobDefinitionParameters(jobDefinitionName);
```

OR
If you want to change the job definition parameters during run time, then add the following code;

```
string jobDefinitionName = "job-definition-name";

// Must start with a '\'
var rootDirectories = new List<string> {@"\root"};

// Name of the volume on the StorSimple device.
var volumeNames = new List<string> {"volume-name"};

var dataTransformationInput = new DataTransformationInput
{
        // If you require the latest existing backup to be picked else use TakeNow to
        trigger a new backup.
        BackupChoice = BackupChoice.UseExistingLatest.ToString(),
        // Name of the StorSimple device.
        DeviceName = "device-name",
        // Name of the container in Azure storage where the files will be placed after
        execution.
        ContainerName = "container-name",
        // File name filter (search pattern) to be applied on files under the root
        directory. * - Match all files.
        FileNameFilter = "*",
        // List of root directories.
        RootDirectories = rootDirectories,
        // Name of the volume on StorSimple device on which the relevant data is present.
        VolumeNames = volumeNames
};
```

7. After the initialization, add the following code to trigger a data transformation job on the job definition, plug in the appropriate Job Definition Name.

```
// Trigger a job, retrieve the jobId and the retry interval for polling.
int retryAfter;
string jobId = dataTransformationJob.RunJobAsync(jobDefinitionName,
    dataTransformationInput, out retryAfter);
```

8. This job will upload the matched files present under the root directory on the StorSimple volume to the specified container. As soon as a file is uploaded, a message is dropped in the Queue (in the same storage account as the container) with the same name as the job definition. This message can be used as a trigger to initiate any further processing of the file.

9. Once the job has been triggered, add the following code to track the job for completion.

```
Job jobDetails = null;

// Poll the job.
do
{
    jobDetails = dataTransformationJob.GetJob(jobDefinitionName, jobId);

    // Wait before polling for the status again.
    Thread.Sleep(TimeSpan.FromSeconds(retryAfter));

} while (jobDetails.Status == JobStatus.InProgress);

// Completion status of the job.
Console.WriteLine("JobStatus: {0}", jobDetails.Status);

// To hold the console before exiting.
Console.Read();
```

Summary

This document showed how you can use the StorSimple Data Transformation service within the StorSimple Data Manager to use data that you have on your device with other Azure services in the cloud.