paolosalvatori

# How to use the Azure Data Lake Store output of an Azure Stream Analytics job in an IoT scenario

This sample shows how to use the Azure Data Lake Store output to store the results produced by an **Azure Stream Analytics** job that processes telemetry data generated by a range of remote devices. This sample also demonstrates how to use the **Azure Data Lake Store .NET SDK** to build a client application that allows a user to navigate and read data an **Azure Data Lake Store** account.

# Scenario

This solution simulates an Internet of Things (IoT) scenario where thousands of devices send events (e.g. sensor readings) to a backend system via a message broker (Event Hubs or IoT Hub). In this context, a **Stream Analytics** job is used to perform the following actions:

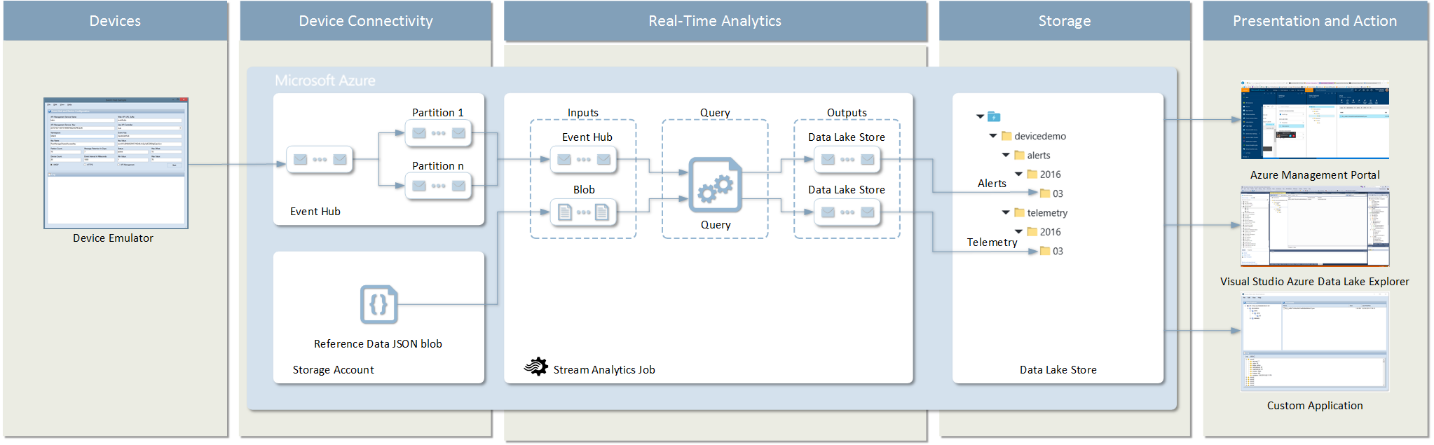
* process device events in a real time fashion
* join telemetry data it with device metadata contained in a JSON file stored in **Blob Storage**
* project enriched data to an Azure Data Lake Store accounts

# Architecture

The sample is structured as follows:

1. A client application can be used to simulate a configurable amount of devices that send readings into the event hub. Each device uses a separate publisher endpoint to send data to an **Event Hub**.
2. An **Event Hub** is used to ingest device events.
3. A **Event Hub** input is used to receive telemetry data from the **Event Hub**, while an **Blob Storage** input is used to read device metadata in **JSON** format from a blob.
4. The job query joins the streaming data from the **Event Hub** with the lookup data contained in the blob and generates two outputs.
   * **Alerts**: a **SELECT** statement checks the value of telemetry data against the minimum and maximum threshold defined on a per device basis and generates a result when the actual value is out of the tolerance range. Alerts are stored as a **JSON Array** in the **devicedemo\* folder inside the specified** Azure Data Lake Store\*\* account using an output called **Alerts**.
   * **Telemetry**: another **SELECT** statement just enriches incoming events with device metadata contained in the blob file. Telemetry data is stored as a **JSON Array** in n the **devicedemo\* folder inside the specified** Azure Data Lake Store\*\* account under a given path using the corresponding **Stream Analytics**.
5. In addition to the **Data Explorer** available in the **Azure Management Portal** and the **Azure Data Lake Store Explorer** tool integrated with **Visual Studio**, the solution provides a client application implemented using **Windows Forms** that allows to navigate and explore results.

The following picture shows the architecture of the solution:



# References

Azure Data Lake Store output for Stream Analytics

* [Stream Analytics Data Lake Store output](https://azure.microsoft.com/en-us/documentation/articles/stream-analytics-data-lake-output/)
* [Target Stream Analytics data transformation outputs to analysis tools and data storage options](https://azure.microsoft.com/en-us/documentation/articles/stream-analytics-define-outputs/)

Azure Data Lake Stream Store .NET SDK  
- [Get started with Azure Data Lake Store using .NET SDK](https://azure.microsoft.com/en-us/documentation/articles/data-lake-store-get-started-net-sdk/)

Event Hubs

* [Event Hubs](http://azure.microsoft.com/en-us/services/event-hubs/)
* [Get started with Event Hubs](http://azure.microsoft.com/en-us/documentation/articles/service-bus-event-hubs-csharp-ephcs-getstarted/)
* [Event Hubs Programming Guide](https://msdn.microsoft.com/en-us/library/azure/dn789972.aspx)
* [Service Bus Event Hubs Getting Started](https://code.msdn.microsoft.com/windowsazure/Service-Bus-Event-Hub-286fd097)
* [Event Hubs Authentication and Security Model Overview](https://msdn.microsoft.com/en-us/library/azure/dn789974.aspx)
* [Service Bus Event Hubs Large Scale Secure Publishing](https://code.msdn.microsoft.com/windowsazure/Service-Bus-Event-Hub-99ce67ab)
* [Service Bus Event Hubs Direct Receivers](https://code.msdn.microsoft.com/windowsazure/Event-Hub-Direct-Receivers-13fa95c6)
* [Service Bus Explorer](https://code.msdn.microsoft.com/windowsapps/Service-Bus-Explorer-f2abca5a)
* [Episode 160: Event Hubs with Elio Damaggio](http://channel9.msdn.com/Shows/Cloud+Cover/Episode-160-Event-Hubs-with-Elio-Damaggio) (video)
* [Telemetry and Data Flow at Hyper-Scale: Azure Event Hub](http://channel9.msdn.com/Events/TechEd/Europe/2014/CDP-B307) (video)
* [Data Pipeline Guidance](https://github.com/mspnp/data-pipeline)  (Patterns & Practices solution)
* [Event Processor Host Best Practices Part 1](http://blogs.msdn.com/b/servicebus/archive/2015/01/16/event-processor-host-best-practices-part-1.aspx)
* [Event Processor Host Best Practices Part 2](http://blogs.msdn.com/b/servicebus/archive/2015/01/21/event-processor-host-best-practices-part-2.aspx)
* [How to create a Service Bus Namespace and an Event Hub using a PowerShell script](http://blogs.msdn.com/b/paolos/archive/2014/12/01/how-to-create-a-service-bus-namespace-and-an-event-hub-using-a-powershell-script.aspx)

# Visual Studio Solution

The Visual Studio solution includes the following projects:

* **Entities**: this library contains the **Payload** class. This class is used by the device simulator and defines the structure and content of the [EventData](https://msdn.microsoft.com/en-us/library/microsoft.servicebus.messaging.eventdata.aspx?f=255&MSPPError=-2147217396) message body.
* **DataLakeExplorer**: this project contains a custom application that uses the **Azure Data Lake Store .NET SDK** library to access telemetry data from the **Azure Data Lake Store** account. This application uses **Azure Active Directiory** NuGet libraries to authenticate the user against an **Azure Active Directory**. Make sure to properly replace the adlsAccount, tenantId, clientId, redirectUrl placeholders with valid values in the configuration file before testing the application. For more information, see [Get started with Azure Data Lake Store using .NET SDK](https://azure.microsoft.com/en-us/documentation/articles/data-lake-store-get-started-net-sdk/).
* **DeviceSimulator**: this **Windows Forms** application can be used to create the **Event Hub** used by the sample and simulate a configurable amount of devices sending telemetry events to the IoT application.

**NOTE**: To reduce the size of the zip file, I deleted the NuGet packages. To repair the solution, make sure to right click the solution and select **Enable NuGet Package Restore**. For more information on this topic, see the following [post](http://blogs.4ward.it/enable-nuget-package-restore-in-visual-studio-and-tfs-2012-rc-to-building-windows-8-metro-apps/).

# Azure Data Lake Store Output

Stream Analytics supports Azure Data Lake Store. This storage enables you to store data of any size, type and ingestion speed for operational and exploratory analytics. At this time, creation and configuration of Data Lake Store outputs is supported only in the Azure Classic Portal. Further, Stream Analytics needs to be authorized to access the Data Lake Store. Details on authorization and how to sign up for the Data Lake Store Preview (if needed) are discussed in the Data Lake output article.

The following list contains the properties needed for creating a Data Lake Store output.

* **Output Alias**: this is a friendly name used in queries to direct the query output to this Data Lake Store.
* **Data Lake Store Account**: the name of the storage account where you are sending your output. You will be presented with a drop down list of Data Lake Store accounts to which the user logged in to the portal has access to.
* **Path Prefix Pattern [optional]**: the file path used to write your files within the specified Data Lake Store Account. Example: *folder1/logs/{date}/{time}*.
* **Date Format [*optional*]**: if the date token is used in the prefix path, you can select the date format in which your files are organized. Example: YYYY/MM/DD
* **Time Format [optional]**: if the time token is used in the prefix path, specify the time format in which your files are organized. Currently the only supported value is HH.
* **Event Serialization Format**: serialization format for output data. JSON, CSV, and Avro are supported.
* **Encoding**: if CSV or JSON format, an encoding must be specified. UTF-8 is the only supported encoding format at this time.
* **Delimiter**: only applicable for CSV serialization. Stream Analytics supports a number of common delimiters for serializing CSV data. Supported values are comma, semicolon, space, tab and vertical bar.
* **Format**: only applicable for JSON serialization. Line separated specifies that the output will be formatted by having each JSON object separated by a new line. Array specifies that the output will be formatted as an array of JSON objects.

# Test Files

The **Test Files** solution folder contains two **JSON** files that you can use to test the **Stream Analytics** job query from the **Azure Management Portal**:

* **DeviceEvents.json**: this file emulates real telemetry data coming through the **Event Hub**.
* **DeviceReferenceData**: this file contains device metadata and it's an integral part of the demo. When testing

## DeviceEvents.Json

[  
 {  
 "deviceId":1,  
 "name":"device001",  
 "value": 5,  
 "status": "active",  
 "timestamp": "2015-12-16T10:00:00.0000000Z"  
 },  
 {  
 "deviceId":3,  
 "name":"device003",  
 "value":23,  
 "status": "active",  
 "timestamp": "2015-12-16T10:00:05.0000000Z"  
 },  
 {  
 "deviceId":5,  
 "name":"device005",  
 "value":17,  
 "status": "active",  
 "timestamp": "2015-12-16T10:00:20.0000000Z"  
 },  
 {  
 "deviceId":1,  
 "name":"device001",  
 "value":8,  
 "status": "active",  
 "timestamp": "2015-12-16T10:00:30.0000000Z"  
 },  
 {  
 "deviceId":3,  
 "name":"device003",  
 "value":22,  
 "status": "active",  
 "timestamp": "2015-12-16T10:00:35.0000000Z"  
 },  
 {  
 "deviceId":4,  
 "name":"device004",  
 "value": 18,  
 "status": "active",  
 "timestamp": "2015-12-16T10:00:36.0000000Z"  
 },  
 {  
 "deviceId":1,  
 "name":"device001",  
 "value":65,  
 "status": "active",  
 "timestamp": "2015-12-16T10:01:00.0000000Z"  
 },  
 {  
 "deviceId":3,  
 "name":"device003",  
 "value":28,  
 "status": "active",  
 "timestamp": "2015-12-16T10:01:05.0000000Z"  
 },  
 {  
 "deviceId":5,  
 "name":"device005",  
 "value":3,  
 "status": "active",  
 "timestamp": "2015-12-16T10:01:15.0000000Z"  
 },  
 {  
 "deviceId":1,  
 "name":"device001",  
 "value":54,  
 "status": "active",  
 "timestamp": "2015-12-16T10:01:30.0000000Z"  
 },  
 {  
 "deviceId":3,  
 "name":"device003",  
 "value":43,  
 "status": "active",  
 "timestamp": "2015-12-16T10:01:35.0000000Z"  
 }  
 ]

## DeviceReferenceData.json

This file needs to be copied to **Blob Storage** and used as reference data. It contains a record for each device.

[  
 {  
 "deviceId": 1,  
 "location": "Milan",  
 "building": "A01",  
 "minThreshold": 20.0,  
 "maxThreshold": 50.0  
 },  
 {  
 "deviceId": 2,  
 "location": "Milan",  
 "building": "A01",  
 "minThreshold": 20.0,  
 "maxThreshold": 50.0  
 },  
 {  
 "deviceId": 3,  
 "location": "Milan",  
 "building": "A01",  
 "minThreshold": 20.0,  
 "maxThreshold": 50.0  
 },  
 {  
 "deviceId": 4,  
 "location": "Milan",  
 "building": "A01",  
 "minThreshold": 20.0,  
 "maxThreshold": 50.0  
 },  
 {  
 "deviceId": 5,  
 "location": "Milan",  
 "building": "A02",  
 "minThreshold": 20.0,  
 "maxThreshold": 40.0  
 },  
 ...  
 ]

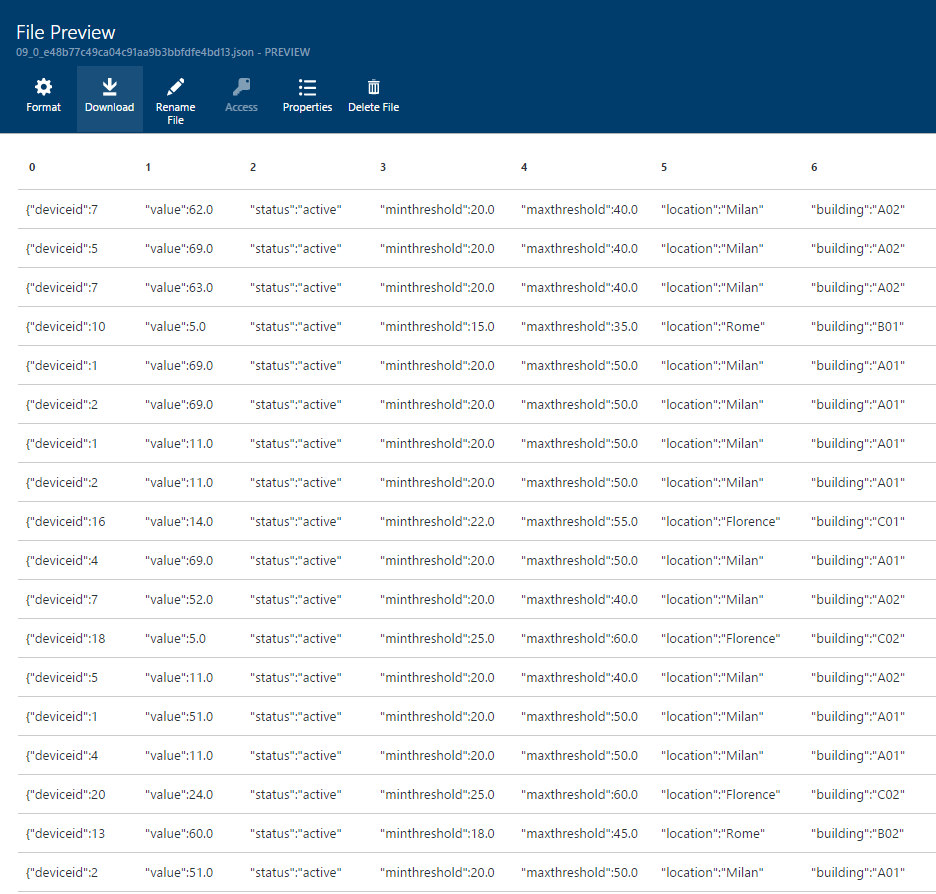
# Query

The following table contains the code of the query used by the **Streaming Analytics** job to join telemetry data with metadata contained in the blob storage and store, respectively, alerts and enriched telemetry data to **Azure Data Lake Store** using two separate outputs.

SELECT E.DeviceId,   
 E.Value,   
 E.Status,   
 R.MinThreshold,   
 R.MaxThreshold,   
 R.Location,   
 R.Building,   
 System.TimeStamp as EntryTime   
 INTO Alerts  
 FROM DeviceEvents E  
 JOIN DeviceReferenceData R  
 ON E.DeviceId = R.DeviceId  
 WHERE E.Value < R.MinThreshold  
 OR E.Value > R.MaxThreshold   
   
 SELECT E.DeviceId,   
 E.Value,   
 E.Status,   
 R.MinThreshold,   
 R.MaxThreshold,   
 R.Location,   
 R.Building,   
 System.TimeStamp as EntryTime  
 INTO Telemetry  
 FROM DeviceEvents E  
 JOIN DeviceReferenceData R  
 ON E.DeviceId = R.DeviceId

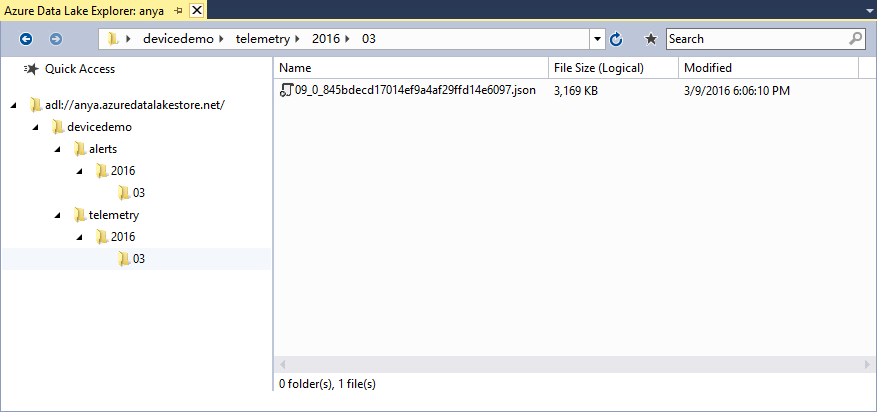
# Data Explorer in the Azure Management Portal

The following image shows how you can use **Data Explorer** integrated in the **Azure Management Portal** to display results.



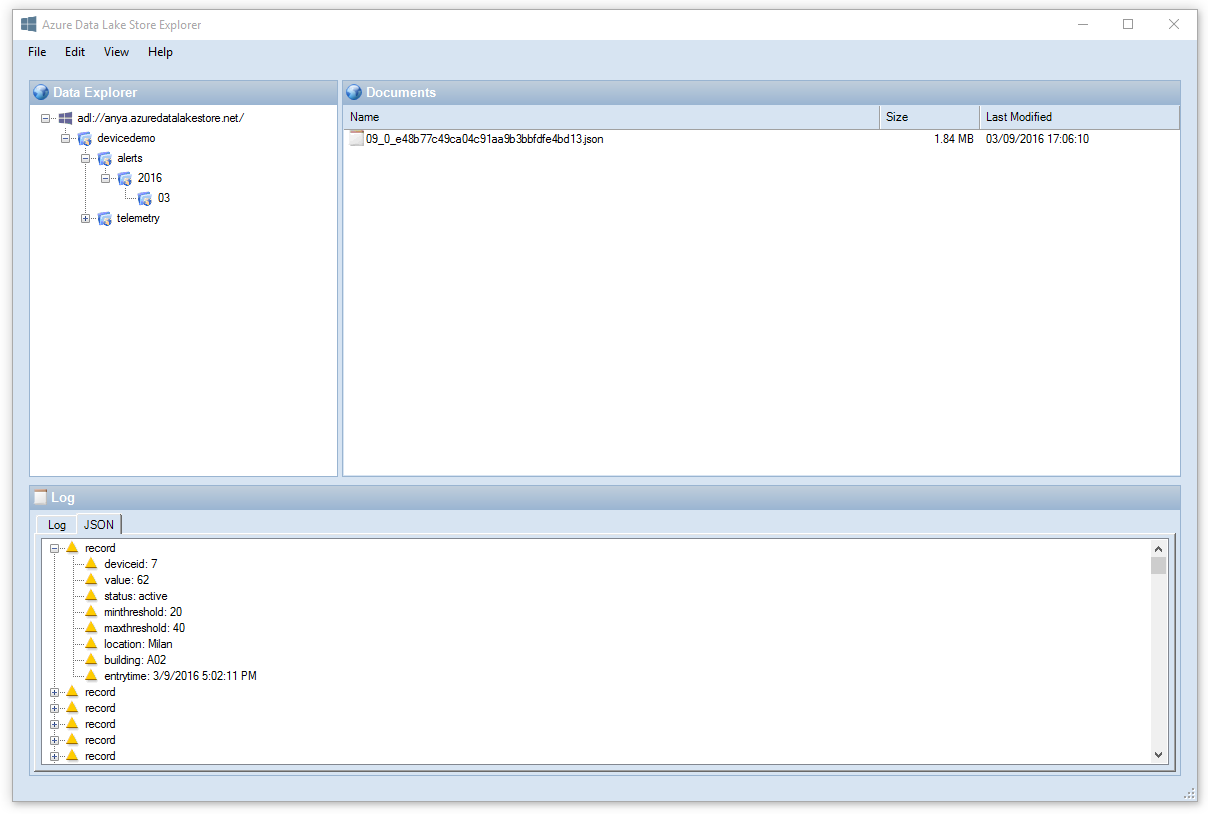
# Azure Data Lake Store Explorer in Visual Studio

The following image shows how you can use **Azure Data Lake Store Explorer** in **Visual Studio** to browse and access data stored in your **Azure Data Lake Store** account.



# Azure Data Lake Store Explorer

This application can be used to navigate and read the data stored in a **JSON Array** format in any file contained in an **Azure Data Lake Store** account. This application represents the embryo of a general purpose **Azure Data Lake Store Explorer**.



The following table contains a snippet from the configuration file. Make sure to replace the placeholders in the **appSettings** section with valid values before testing the application. For more information, see [Get started with Azure Data Lake Store using .NET SDK](https://azure.microsoft.com/en-us/documentation/articles/data-lake-store-get-started-net-sdk/).

<?xml version="1.0" encoding="utf-8"?>  
 <configuration>  
 <appSettings>  
 <add key="adlsAccountName" value="[AZURE-DATA-LAKE-STORE-ACOOUNT]"/>  
 <add key="subscriptionId" value="[AZURE-SUBSCRIPTION-ID]"/>  
 <add key="tenantId" value="[TENANT-ID]"/>  
 <add key="clientId" value="[CLIENT-ID]"/>  
 <add key="redirectUri" value="[REDIRECT-URL]"/>  
 </appSettings>  
 <startup>  
 <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />  
 </startup>  
 ...  
 </configuration>