Lab: Enabling a Real-Time Microsoft Power BI dashboard with Azure Stream Analytics

Contents

Overview

Summary

In this lab we are going to learn how to build real-time dashboards in Power BI using Azure Stream Analytics (ASA). This lab takes about 30 minutes to complete.

Scenario

You will build a real-time dashboard that shows the data generated by the sensors on your device. Also, you will add a threshold value. You will be using ASA to analyze the sensor data from IoT hub and send alerts to a Power BI dashboard.

You will take the following steps:

1) Create an Azure Stream Analytics job
2) Querying the stream
3) Create a Power BI Dashboard

Azure Stream Analytics Summary

Stream Analytics is a fully managed service providing low-latency, highly available, scalable complex event processing over streaming data in the cloud. Stream Analytics makes it easy to set up real-time

analytic computations on data streaming from devices, sensors, web sites, social media, applications, infrastructure systems, and more.

With a few clicks in the Azure portal, you can author a Stream Analytics job specifying the input source of the streaming data, the output sink for the results of your job, and a data transformation expressed in a SQL-like language. You can monitor and adjust the scale/speed of your job in the Azure portal to scale from a few kilobytes to a gigabyte or more of events processed per second.

Stream Analytics leverages years of Microsoft Research work in developing highly tuned streaming engines for time-sensitive processing, as well as language integrations for intuitive specifications of such.

What can I use Stream Analytics for?

Vast amounts of data are flowing at high velocity over the wire today. Organizations that can process and act on this streaming data in real time can dramatically improve efficiencies and differentiate themselves in the market. Scenarios of real-time streaming analytics can be found across all industries:

personalized, real-time stock-trading analysis and alerts offered by financial services companies; real-time fraud detection; data and identity protection services; reliable ingestion and analysis of data generated by sensors and actuators embedded in physical objects (Internet of Things, or IoT); web clickstream analytics; and customer relationship management (CRM) applications issuing alerts when customer experience within a time frame is degraded. Businesses are looking for the most flexible, reliable and costeffective way to do such real-time event-stream data analysis to succeed in the highly competitive modern business world.

Key capabilities and Benefits?

- **Ease of use:** Stream Analytics supports a simple, declarative query model for describing transformations. In order to optimize for ease of use, Stream Analytics uses a SQL variant, and removes the need for customers to deal with the technical complexities of stream processing systems. Using the Stream Analytics query language in the browser query editor, you get intellisense autocomplete to help you can quickly and easily implement time series queries, including temporal-based joins, windowed aggregates, temporal filters, and other common operations such as joins, aggregates, projections, and filters. In addition, in-browser query testing against a sample data file enables quick, iterative development.

- **Scalability:** Stream Analytics is capable of handling high event throughput of up to 1GB/second. Integration with Azure Event Hubs allows the solution to ingest millions of events per second coming from connected devices, clickstreams, and log files, to name a few. In order to achieve this, Stream Analytics leverages the partitioning capability of Event Hubs, which can yield 1MB/s per partition. Users are able to partition the computation into a number of logical steps within the query definition, each with the ability to be further partitioned to increase scalability.

- **Reliability, repeatability and quick recovery:** A managed service in the cloud, Stream Analytics helps prevent data loss and provides business continuity in the presence of failures through built-in recovery capabilities. With the ability to internally maintain

state, the service provides repeatable results ensuring it is possible to archive events and reapply processing in the future, always getting the same results. This enables customers to go back in time and investigate computations when doing root-cause analysis, what-if analysis, etc.

- **Low cost:** As a cloud service, Stream Analytics is optimized to provide users a very low cost to get going and maintain real-time analytics solutions. The service is built for you to pay as you go based on Streaming Unit usage and the amount of data processed by the system. Usage is derived based on the volume of events processed and the amount of compute power provisioned within the cluster to handle the respective Stream Analytics jobs.

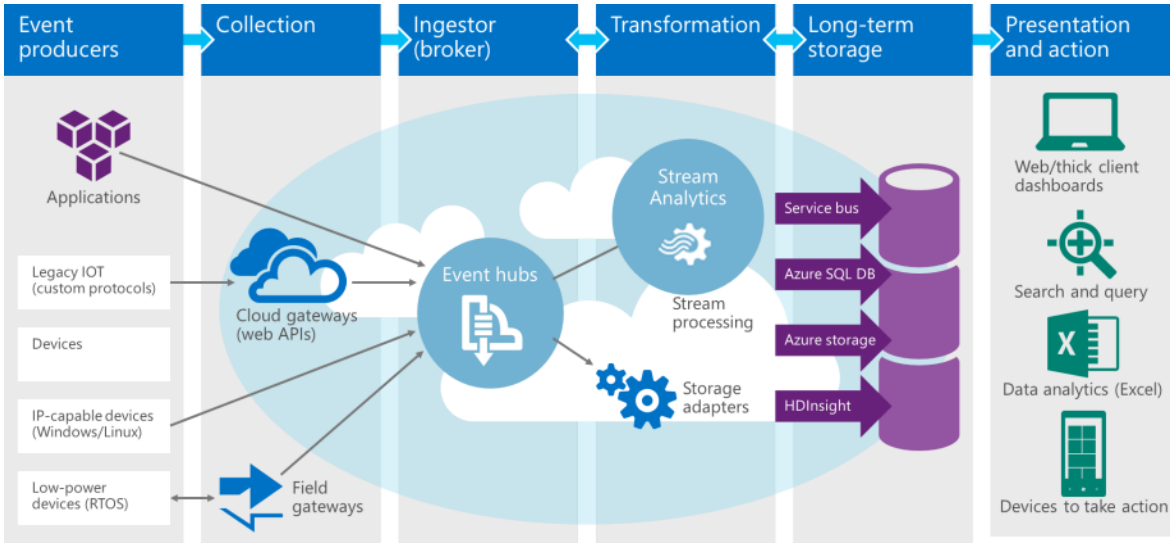
- **Reference data:** Stream Analytics provides users the ability to specify and use reference data. This could be historical data or simply non-streaming data that changes less frequently over time. The system simplifies the use of reference data to be treated like any other incoming event stream to join with other event streams ingested in real time to perform transformations.

- **Connectivity:** Stream Analytics connects directly to Azure Event Hubs for stream ingestion, and the Azure Blob service to ingest historical data. Results can be written from Stream Analytics to Azure Storage Blobs or Tables, Azure SQL DB, Event Hubs, Azure Service Bus Topics or Queues, and Power BI, where it can then be visualized, further processed by workflows, used in batch analytics via Azure HDInsight or processed again as a series of events. When using Event Hubs it is possible to compose multiple Stream Analytics together with other data sources and processing engines without losing the streaming nature of the computations.

End-to-end stream processing on Microsoft Azure

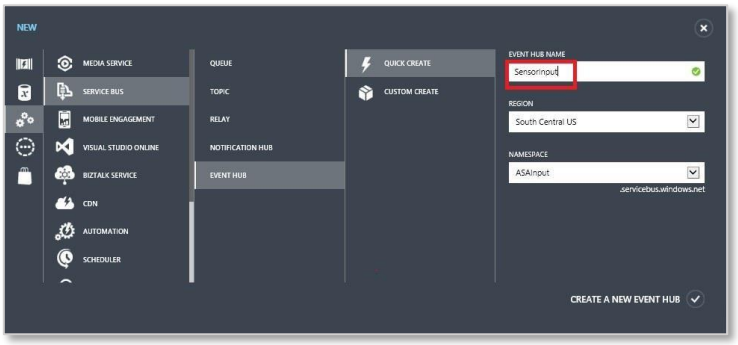
Applications or devices produce messages which are sent to an *Azure Event Hub*. Event Hubs acts as the "front door" for an event pipeline, and once data is collected into an event hub, it can be transformed and stored using any real-time analytics provider or batching/storage adapters. Event Hubs decouples the production of a stream of events from the consumption of those events, so that event consumers can access the events on their own schedule. Unlike Service Bus queues and topics, Event Hubs is focused on delivering messaging stream handling at scale. Event Hubs capabilities differ from topics in that they are strongly biased towards high throughput and event processing scenarios. As a result, Event Hubs do not implement some of the messaging capabilities that are available for topics. If you need those capabilities, topics remain the optimal choice.

Stream Analytics is the *consumer* of the event hub, which queries the data in the Event Hub (in real-time) using a SQL-like language and produces an output sink to a service bus, blob storage, Azure SQL DB or HDInsight



This allows the output of the Stream Analytics query to be presented in PowerBI, Web dashboards or in devices to take action.

Step 1: Create Stream Analytics Job



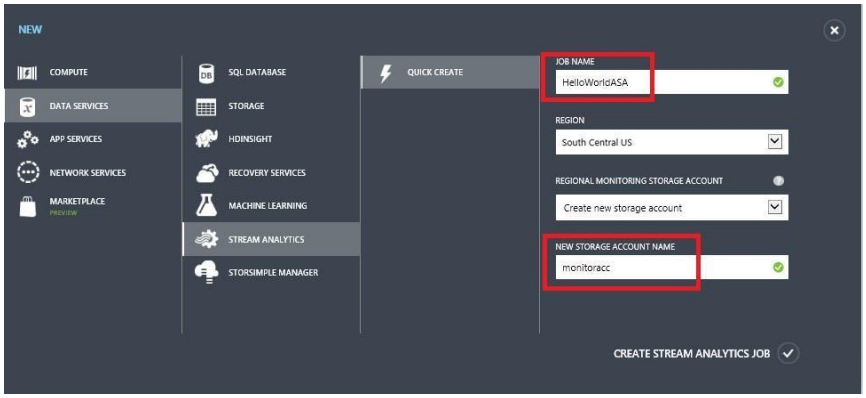
1. In Azure portal open **Stream Analytics** and click **New** in the bottom left hand corner of the page to create a new analytics job.

2. Click **Quick Create**.

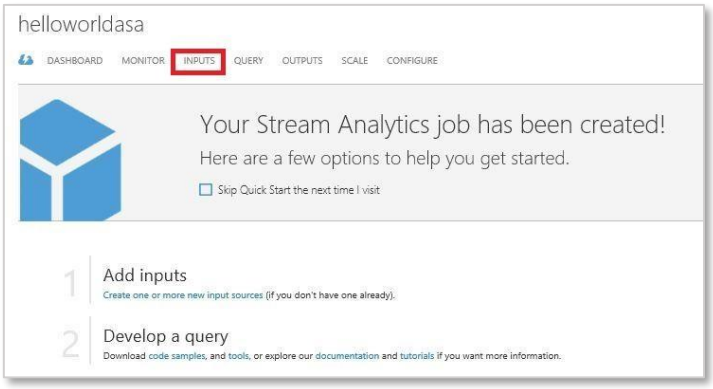
3. For **Regional Monitoring Storage Account** setting, select **Create new storage account** and give it any unique name. Azure Stream Analytics will use this account to store monitoring information for all your future jobs.

   Note: You should create this storage account only once per region and this storage will be shared across all ASA jobs created in that region.

4. Click **Create Stream Analytics Job** at the bottom of the page.

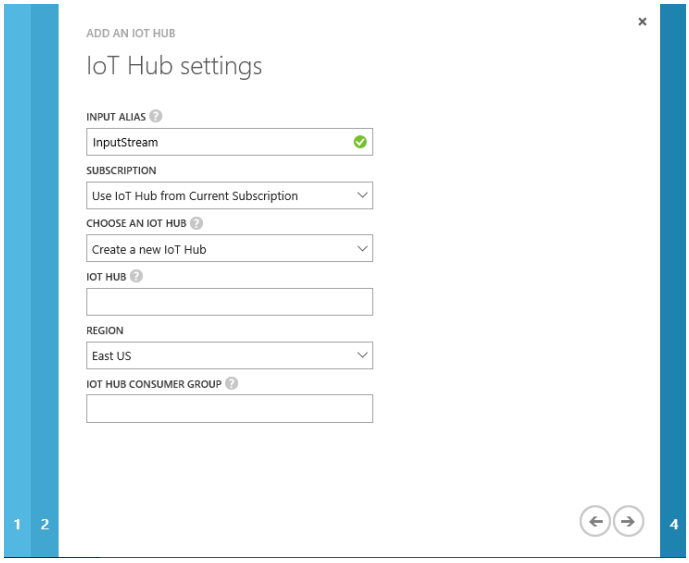


5. Click on the created analytics job in the portal.
6. Open **Inputs** tab to define the source data.



7. Click **Add an Input**. Select **Data Stream** on the first page. Select **IoT Hub** on the second page of the wizard and configure it to link to your IoT Hub. Enter **InputStream** as Input Alias.

   Your settings will look like this:



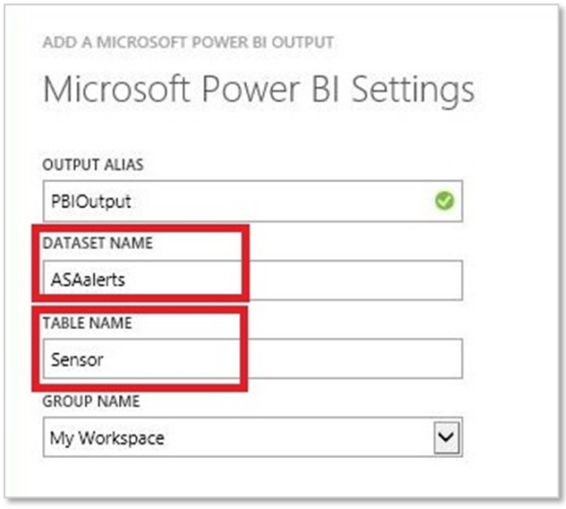
8. Move to the next page. Select values as **JSON**, **UTF8** encoding.

9. Go to **Output** tab and click **Add an output**.

10. We support a list of different outputs. Select **Power BI** from the list.

11. In this step, you will need a work or school account for the Stream Analytics job output. If you already have Power BI account, select **Authorize Now**. If not, choose **Sign up now**.

12. Add Name of the Dataset and the table for Power BI.



Note:
You should not explicitly create this dataset and table in your Power BI account. They will be automatically created when you start your Stream Analytics job and the job starts pumping output into Power BI. If your job query doesn't return any results, the dataset and table will not be created.

WARNING:
Be aware that if Power BI already had a dataset and table with the same name as the one you provided in this Stream Analytics job, the existing data will be overwritten.

13. Once output is added you should see it in your list of output.

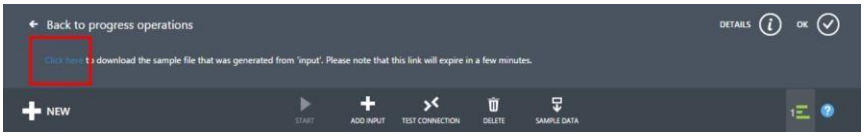

helloworldasa
DASHBOARD
MONITOR
INPUTS
QUERY
OUTPUTS
SCALE
CONFIGURE
NAME
SINK
DIAGNOSIS
PBIOutput
Power BI
OK

Step 2: Querying the stream

Stream Analytics supports a simple, declarative query model for describing transformations for real-time processing. To learn more about the language, see the Azure Stream Analytics Query Language Reference. This tutorial will help you author and test several queries over your real-time stream of call data.

To validate your query against actual job data, you can use the **Sample Data** feature to extract events from your stream and create a .JSON file of the events for testing. The following steps show how to do this:

1. Select your Event Hub input and click **Sample Data** at the bottom of the page.
2. In the dialog box that appears, specify a **Start Time** to start collecting data from and a **Duration** for how much additional data to consume.
3. Click the check button to start sampling data from the input. It can take a minute or two for the data file to be produced. When the process is completed, click **Details** and download and save the .JSON file that is generated.

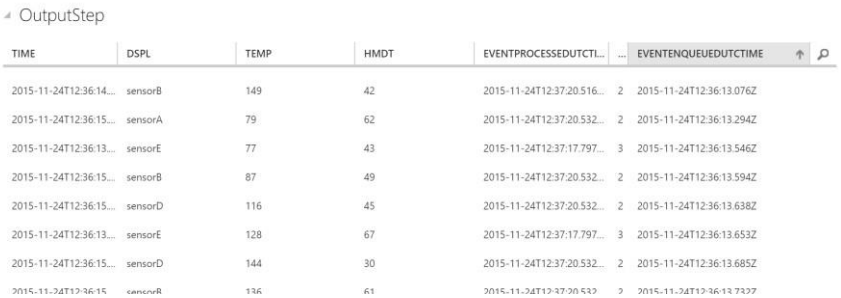


4. If you want to archive every event, you can use a **pass through query** to read all the fields in the payload of the event or message. To start with, do a simple pass through query that projects all the fields in an event.
5. Click **Query** from the top of the Stream Analytics job page.
6. Add the following to the code editor:

```
SELECT * FROM InputStream
```

Make sure that the name of the input source matches the name of the input you specified earlier.

7. Click **Test** under the query editor.
8. Supply a test file – the one that you created using the previous steps.
9. Click the check button and see the results displayed below the query definition.

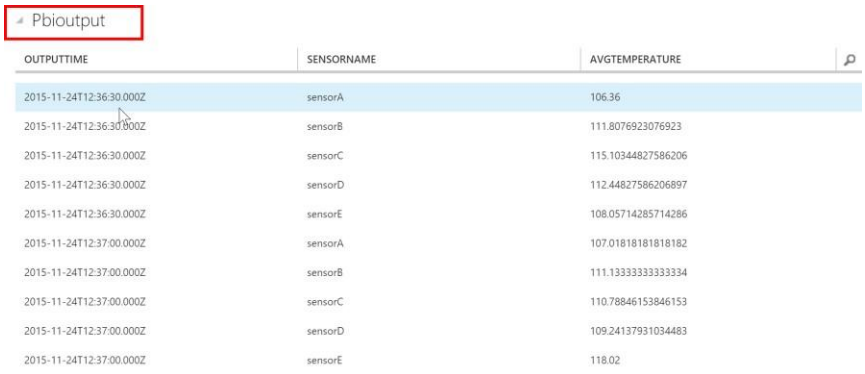


We'll now pare down the returned fields to a smaller set.

10. Change the query in the code editor to:

```
SELECT
CAST(time as datetime) as time,
CAST(temperature as float) as temperature,
CAST(lumen as float) as lumen,
20 as baselinetemperature
INTO PBIOutput
FROM InputStream
```

11. Click **Rerun** under the query editor to see the results of the query.



12. **Save** the query.

13. **Start** the Stream Analytics job (keep the default **Job Start Time**).

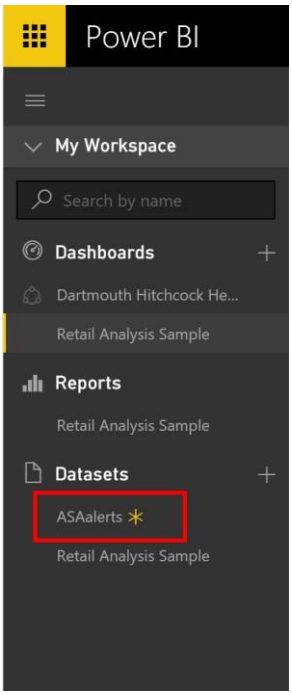
Step 3: Create PowerBI dashboard

Now that we have defined an event stream, an Event Hub input to ingest events, and a query to perform a transformation over the stream, the last step is to create a PowerBI Dashboard.

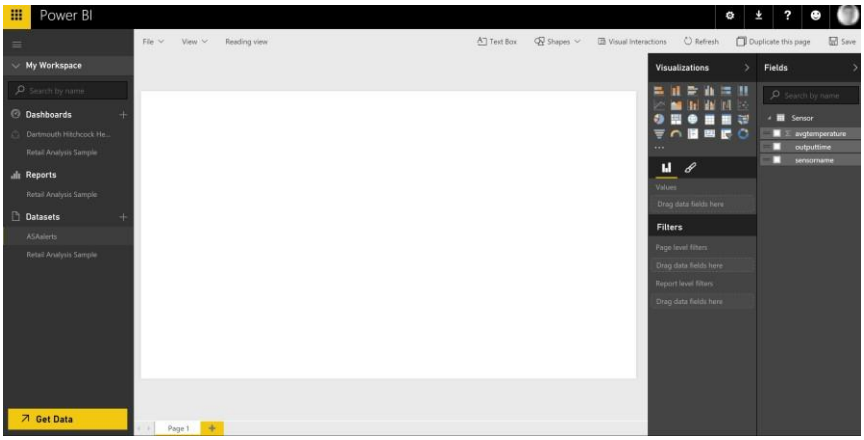
Follow the steps below:

1. Log in to PowerBI - https://powerbi.microsoft.com/en-us

2. On the landing page you should see the ASAalerts output dataset that we created in the previous section:



3. Click on the ASAalerts Dataset – you should see the following:

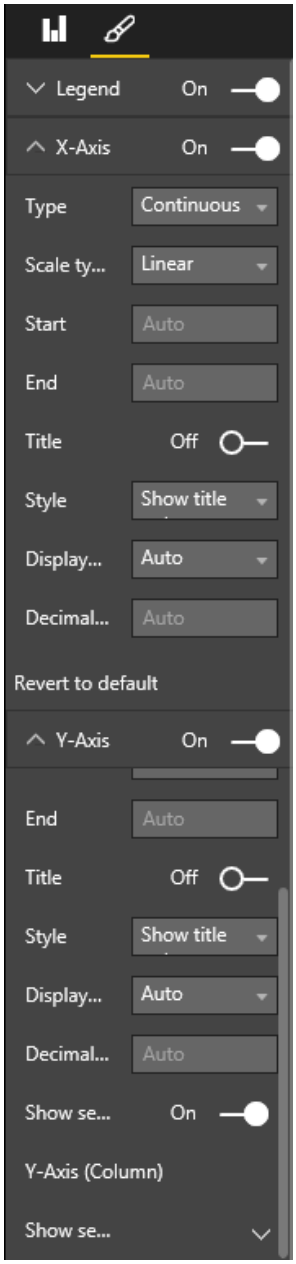


4. Click on the **Line and clustered column Chart** (second row, fifth item) icon in the Visualisations blade.

5. Drag **time** to the shared axis field of the chart.

6. Drag the **lumen** and **temperature** values to the column values field. Drag the **baselinetemperature** field to the line values field.

7. Change the scale of the second axis by going to the layout options (with the chart selected click the pencil icon) and open the Y-Axis blade. Toggle the secondary axis by setting **Show Secondary Axis** to on. You might need to change start and end values of the axis to make sure the scale is OK.



8. Click on the **Pin** > Create a new Report > Pin to dashboard (select a name for a **New Dashboard**).

9. Click on the Dashboard you just created and you should see your chart updating every 10 seconds.

10. Explore

    - other visualisations
    - tables
    - natural language search feature