TR24 Smart Spaces Hands-on Lab

Welcome to the TR24 Smart Spaces Hands-on Lab!

We put together this workshop to help you understand the basic constructs of a Building Management System using only Microsoft technologies. You will build an end-to-end solution that pushes data from one or more simulated building sensors to Azure and then runs some basic data processing and analytics on that data, in the hope that this will give you a clearer understanding of what commercial BMSs like ICONICS do. The working solution is NOT meant to be a replacement for an enterprise BMS, but only a Proof of Concept.

Assumptions and Prerequisites

In order to complete this lab without seriously struggling, you need to have a good understanding of Microsoft Azure. Specifically, you should have done each of the following more than a few times:

- Used http://portal.azure.com
- Created an IoT Hub or Event Hub
- Used Device Explorer or other tool to view data coming into IoT Hub
- Created an Azure Stream Analytics job
- Created and used Azure blob storage
- Created tables, views, and stored procedures in SQL Azure

An optional component of this lab uses Visual Studio to compile and run a program. A workshop participant wishing to do this part of the lab needs to have the following installed on his/her laptop:

- Git Shell or Git for Windows
- Visual Studio 2013 or later

A participant doing this part of the lab would also need familiarity with:

- Accessing http://GitHub.com
- Using Visual Studio to build and run a Project

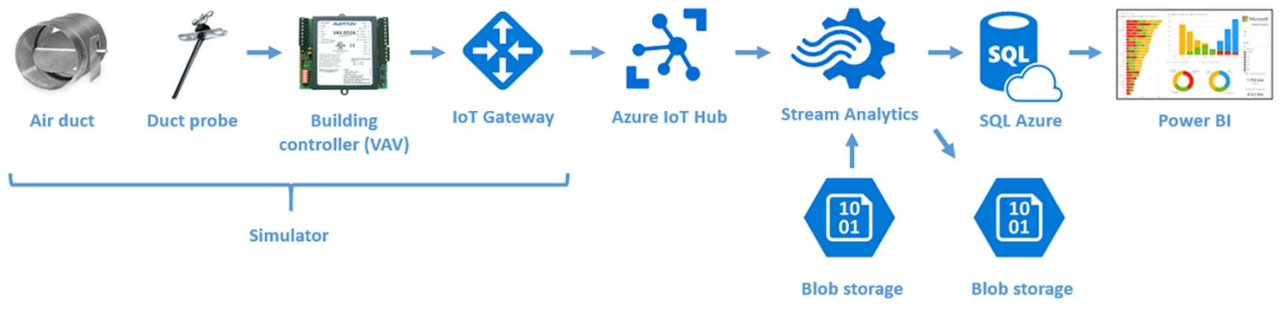
Getting Started

Each workshop participant has been provided access to the Azure subscription for the lab, and will be building assets in that subscription. In addition, we have already created some objects that every participant will be using. Here are the names you need to know:

Azure subscription ID	Microsoft Azure Internal Consumption (da1dbdc2-....)
IoT Hub:	TR24SmartBuildingHub
SQL Server	tr24smartbuilding.database.windows.net
SQL Database	tr24SmartBuilding
SQL account	TR24_Admin
SQL password	PCS2017!
Azure blob storage account	tr24smartbuilding
Device map file	refdata/bacmap/2017/.../.../bacmap.csv

Whenever you encounter an instruction in the lab to create something, you should preface the name of the object you are creating with your initials. For example, if your alias is Spyros and are asked to create an Azure Stream Analytics job called {Alias}LogAllEvents you would create a job with the name SpyrosLogAllEvents.

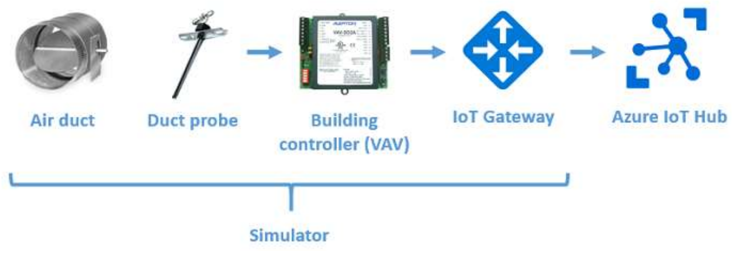
Here is a picture of this step in the end-to-end solution:



In this workshop we will build this end-to-end solution in stages.

Exercise 1: Load a device simulator

Here is a picture of this step in the end-to-end solution:



Please follow these steps:

1. Logged into your laptop using your Microsoft credentials, access the private Smart Building repository on the Azure-Samples website. Either download SmartBuildingSimulatorBinaries.zip from the/Simulator/binaries folder and unzip it, or, optionally, use Git to clone the entire project to your laptop and build SimulatedSensors.Windows.sln using Visual Studio.
2. Read through the readme.md file in the Simulator folder to get an understanding of what the Simulator does.
3. Launch SimulatedSensors.Windows.exe and follow the steps in the readme.md file to
   - Get the connection string for TR24SmartBuildingHub
   - Get the connection string for TR24SmartBuildingDB
   - Find a DeviceId registered in the IoT Hub
   - Select names for the simulator text fields from the drop-down menus
   - Send data and verify it is being received by the IoT Hub

   When prompted to enter data in the Simulator, use the following:

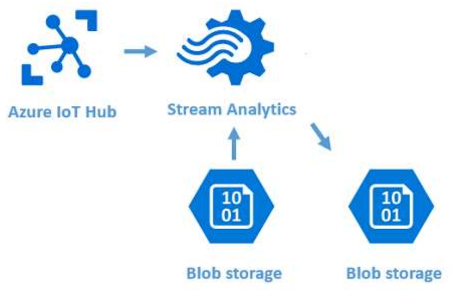
- **IoT Connection String**: {TR24SmartBuildingHub Connection String}
- **SQL Connection String**: {TR24SmartBuildingDB Connection String}
- **GatewayName**: Select {Alias}Gateway from the drop-down list of available GatewayNames
- **DeviceName**: Select from drop-down list
- **ObjectType**: Select from drop-down list
- **Instance**: Select from drop-down list

You have successfully completed this exercise if and only if the data that you see being transmitted from the Simulator is included in the data you see being received in Device Explorer. Bear in mind that the data from all the other workshop participants is also being received by the same IoT Hub, so you will need to watch carefully for your data.

Estimated time to complete: 15 minutes

Exercise 2: Create an Azure Stream Analytics job

In this exercise, you will push data from the device into blob storage. Here is a picture of this step in the end-to-end solution:



Please follow these steps:

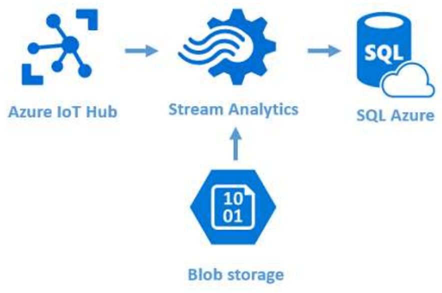
1. Download and open BACmap.csv to view the reference data that you need to join with the streaming device data in IoT Hub.
2. Follow the steps in the readme.md file in the Azure/StreamAnalytics folder of the Azure-Samples website to create an Azure Stream Analytics job called LogAllEvents that outputs the data received from IoT hub for your Simulator to blob storage
   - Join the data from the Simulator with the reference data in BACmap.csv. Don't forget to add a filter in the query to capture only your Simulator's data.
   - Output the data to a blob called {Alias}ASAOutput in the container tr24smartbuilding
3. Download the output blob to verify that you have correctly captured only the data from your Simulator.

You have successfully completed this exercise if and only if the data that you saw in Device Explorer is represented in the output blob, but instead of having the BACnet addressing scheme for your simulated device, it has the physical address of the device.

Estimated time to complete: 20 minutes

Exercise 3: Push the device data into an Event Historian table in SQL

In this exercise, you will modify the Stream Analytics job to push data from your device to a table in SQL Azure. Here is a picture of this step in the end-to-end solution:



Please follow these steps:

1. Read through the readme.md file in the Azure/SQLServer/EventHistorian folder to get an understanding of what the EventHistorian is used for.
2. Edit and execute the dbo.EventHistorian.Table.sql script in the Azure/SQLServer/EventHistorian folder to create the SQL table dbo.{Alias}EventHistorian in the tr24SmartBuilding database.
3. Modify the LogAllEvents Stream Analytics job to output data to dbo.{Alias}EventHistorian instead of the {Alias}ASAOutput blob
4. Send more data from the simulator to IoT Hub. Try changing the device and object names to send different values to the Events table.
5. List the records in dbo.{Alias}EventHistorian.

You have successfully completed this exercise if and only if the data that you saw in {Alias}ASAOutput blob are represented in the dbo.{Alias}EventHistorian listing.

Estimated time to complete: 10 minutes

Exercise 4: Send the event history to a pivot table

In this exercise, you will create and execute a SQL Stored Procedure to take the raw event data and pivot into a format for easier consumption by Power BI. Physical devices will typically have many different sensors emitting data (for example air flow, input and output temperatures), and the pivot would be more populated than you will see when sending a single sensor reading from your simulator, but we need to create the pivot files for use in subsequent exercises. In this step in creation of the end-to-end solution we will be working only in SQL Azure:

Please follow these steps:

1. Read through the readme.md file in the Azure/SQLServer/EventProcessing folder to get an understanding of what the stored procedure is doing.
2. Edit and execute the dbo.Pivot_VAV.Table.sql script in the Azure/SQLServer/EventProcessing folder to create the SQL table dbo.{Alias}Pivot_VAV in the tr24SmartBuilding database.
3. Edit and execute the dbo.sp_PopulateEquipmentPivots.StoredProcedure stored procedure to push records from dbo.{Alias}EventHistorian to dbo.{Alias}Pivot_VAV.

4. List the records in dbo.{Alias}Pivot_VAV.
5. Send more records from your simulator, rerun the stored procedure, and verify the records are displayed in the pivot file.

You have successfully completed this exercise if and only if the data that you saw in dbo.{Alias}EventHistorian are represented in dbo.{Alias}Pivot_VAV.

Estimated time to complete: 10 minutes

Exercise 5: Create Fault processing processes

In this exercise, you will create and execute a SQL Stored Procedure to take the pivoted event data and look for data indicative of problems in the equipment, known as 'Faults'. In this step in creation of the end-to-end solution we will again be working only in SQL Azure:

Please follow these steps:

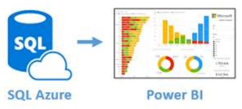
1. Read through the [readme.md]() file in the Azure/SQLServer/FaultProcessing folder to get an understanding of what the stored procedure is doing.
2. Edit and execute the dbo.FaultInstances.Table.sql script in the [Azure/SQLServer/FaultProcessing]() folder to create the SQL table dbo.{Alias}FaultInstances in the tr24SmartBuilding database.
3. Edit and execute the dbo.sp_PopulateFaultInstances.StoredProcedure stored procedure to push records from dbo.{Alias}Pivot_VAV to dbo.{Alias}FaultInstances.
4. List the records in dbo.{Alias}FaultInstances.
5. Send more records from your simulator, but this time increase the temperature to above the threshhold shown in the stored procedure, rerun the two stored procedures, and verify the records are displayed in the fault table.

You have successfully completed this exercise if and only if you are able to generate some faults that you can see in the fault table!

Estimated time to complete: 10 minutes

Exercise 6: Create a Power BI dashboard

In this step in creation of the end-to-end solution we will again be working only in SQL Azure:



Please follow these steps:

- Create a Power BI Dashboard that visualizes your TI Sensor Tag data in creative ways. Feel free to use any of the Power BI Custom Visuals available here. You can learn how to create Power BI Dashboards from a Stream Analytics Output here.

Estimated time to complete: 10 minutes