APPLICATION NOTE: **AN-0344**

AC500 MQTT & MS AZURE

FIRST STEPS AND CONFIGURATION

Contents

1 Abbreviations

AB	Automation Builder
AJAX	Asynchronous JavaScript and XML
CET	Central European Time
CMS	Condition Monitoring System
DTU	Database Throughput Unit[1]
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
MQTT	Message Queuing Telemetry Transport
MS	Microsoft
MVC	Model-Viewer-Controller
PLC	Programmable Logic Controller
RMS	Root Mean Square
SAS	Shared Access Signature
SQL	Standardized Query Language
SSMS	SQL Server Management Studio
TTL	Time-to-live
UTC	Coordinated Universal Time

[1] Measure of performance for databases in MS Azure

2 Introduction

2.1 Scope of the document

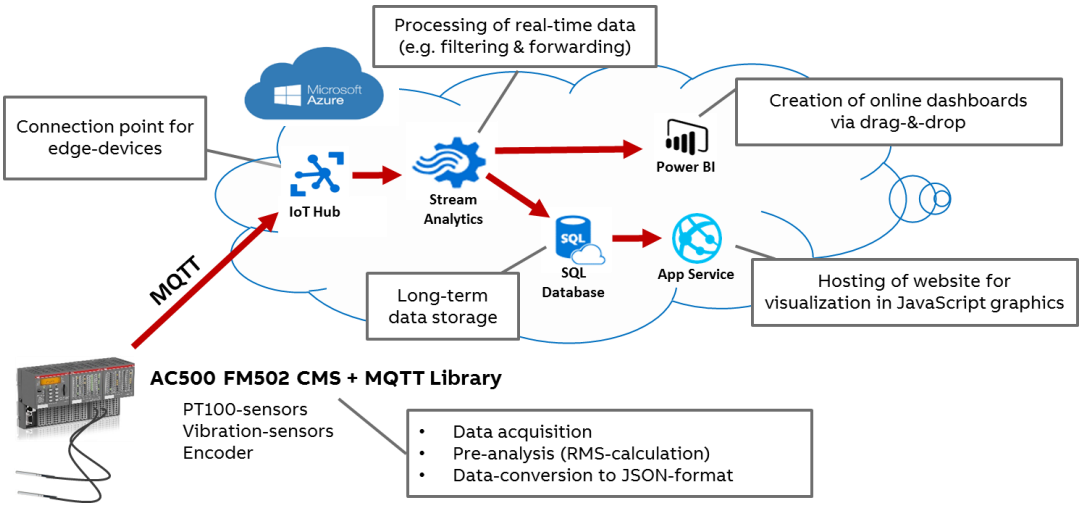
This manual gives a first introduction into setting up a cloud-based condition monitoring system with the AC500 PLC & the MS Azure cloud.
It provides a step-by-step description of cloud configuration for a demo application.

Capabilities of demo application
The demo application is able to store and visualize locally gathered data (temperature, speed, vibration velocity RMS and alarms) in the MS Azure cloud. Therefore, it enables global monitoring of KPIs and alarms.

2.2 Compatibility

The application note explained in this document is using functionalities and screenshots from the MS Azure portal. In future versions the naming or look and feel might be different. Check the azure documentation from Microsoft for further details.

2.3 Overview



Component	Function
AC500 FM502 CMS	Collects data from local sensors (temperature, vibration & speed), processes them (calculation of vibration RMS & alarms, conversion to JSON-format) and sends them to the IoT Hub using MQTT-library
IoT Hub	Manages connection of Edge-devices, receives messages sent from AC500 and makes them accessible for other Azure-services
Stream Analytics	Processes incoming messages in real-time: selects relevant data and forwards them to other Azure services
SQL Database	Receives data from Stream Analytics and stores them
App Service	Hosts website to visualize the data from the database
Power BI	Enables easy creation of online dashboards

For the visualization, two variants will be mentioned:

1. Visualization with a self-created website built inside App Service in MS Azure. The website is created using Asynchronous JavaScript and XML (AJAX) and is based on Model-Viewer-Controller (MVC).

   It is a more sophisticated solution for user who want full control and flexibility of dashboard design and updating intervals. Furthermore, the website can be accessed by anyone who has the link. It does not require special MS licenses. However, it requires additional knowledge and effort of programming the website.

2. Visualization with Power BI, a MS-owned Business Intelligence service. It enables fast creation of dashboards & visuals via drag-and-drop and does not require any programming for visualization. Power BI can be used for free using an MS account. However, to access Power BI dashboards from others or to share own dashboards, users need a Power BI license (Power Bi Pro or Premium).

   Power BI is a good way to get started with cloud visualization as it is easy to use and enables fast creation of dashboards without any programming. However, users are bound to MS and need to have licenses for more advanced functionality.

3 Cloud setup & configuration

3.1 Creation of MS Azure services

To get started, create the following Azure-services:

- IoT Hub
- Stream Analytics
- SQL Database (including server, database and table)
- App Service (For variant 1)

For the second approach, replace App Service by Power BI. This is an additional MS service, but not part of the MS Azure platform.

For further information on how to create and get these services, please follow the links below:

- How to create an IoT Hub
- How to create a Stream Analytics job
- How to create a SQL Database
- App Service overview
- What is Power BI?
- Get started with Power BI

Note: The selection of pricing & scaling models in Azure is depending on the actual requirements for the project

For the **demo application** following is chosen:

IoT Hub

- Subscription: *Choose the Azure subscription*
- Resource group: *Create new group* (Here: “DemoHD”)
- Region: **West Europe**
- IoT Hub name: *Name the IoT Hub* (Here: “DemoHD-IoT-Hub”)
- Pricing tier: **S1: Standard tier**
- No. of IoT Hub units: **1**

Stream Analytics Job

- Job name: *Name the Stream Analytics job*
  (Here: “DemoHD-StreamingAnalytics”)
- Subscription: *Choose the Azure subscription*
- Resource group: *Choose resource group created in IoT Hub*
  (Here: “DemoHD”)
- Location: **West Europe**
- Hosting environment: **Cloud**
- Streaming Units: **3**

SQL-Database

- Database name: *Name the database*
- Select source: **blank database**
- Server: *Create a new server*
  - Server name: *Name the server*
  - Server admin login: *Name the admin login*
  - Password: *Define the password*
  - Server location: **West Europe**
- Elastic pool: **No elastic pool**
- Pricing tier: **Standard S0: 10 DTUs, 250 GB**
- Collation: **SQL_Latin1_General_CP1_CI_AS**
- Location: **West Europe**

Note: For the demo application, the standard pricing tier to have access to the most common functionality (e.g. to enable bidirectional communication with the IoT Hub) is used.

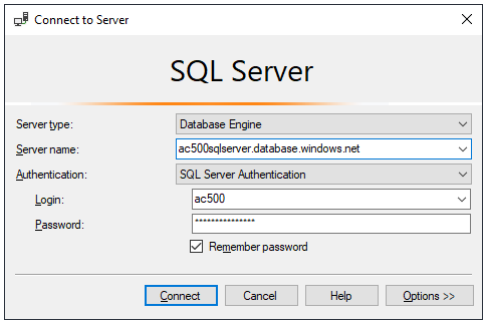
For the scaling (e.g. streaming/ IoT Hub units), the lowest scaling was chosen. Only one device is connecting and no complex data processing on cloud level is done.

3.2 Creation of SQL database tables

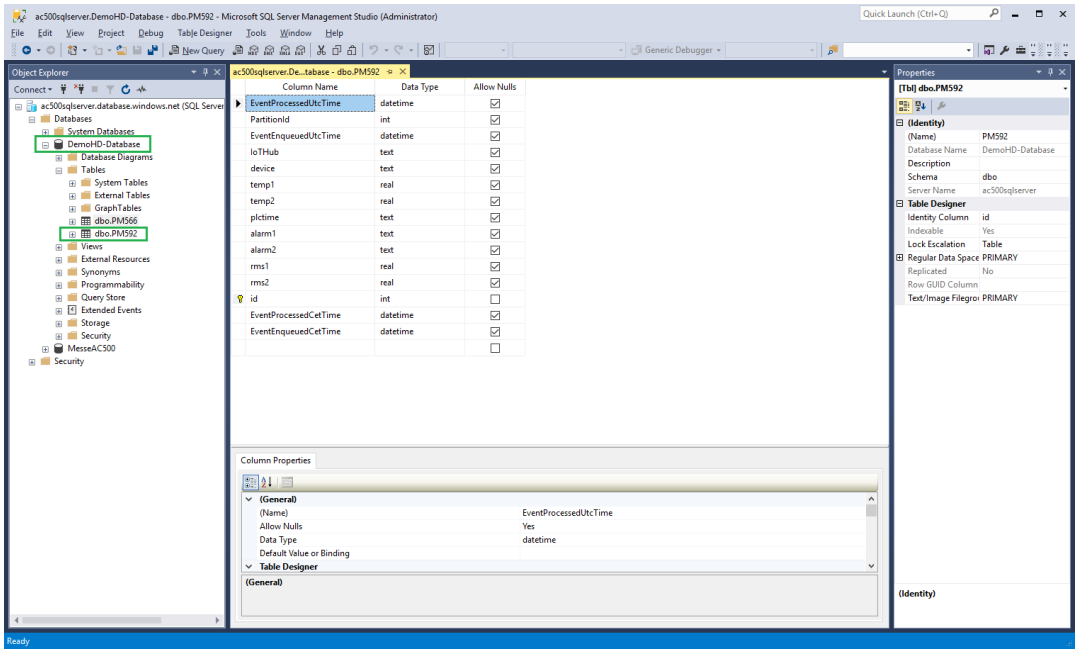
After creation of the SQL Server, “SQL Server Management Studio (SSMS)” can be used to create and modify the SQL databases and tables.

Start the SSMS software:

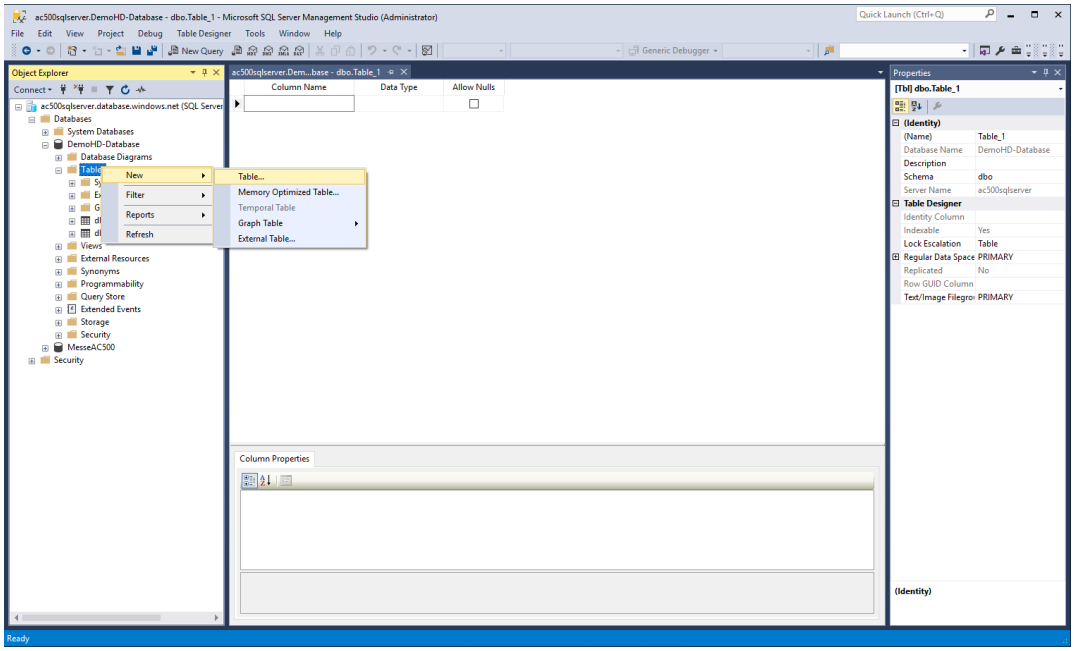
- Add the server name (will be selected when creating the SQL server in Azure)
- Username
- Password



Once logged the defined database, here: “DemoHD-Database” is visible. This database has already two tables. By default, there are no tables visible. We will focus only on table “dbo.PM592” in this example.



If no table like “PM592” is created yet, this can be done by right click - choose New →Table…



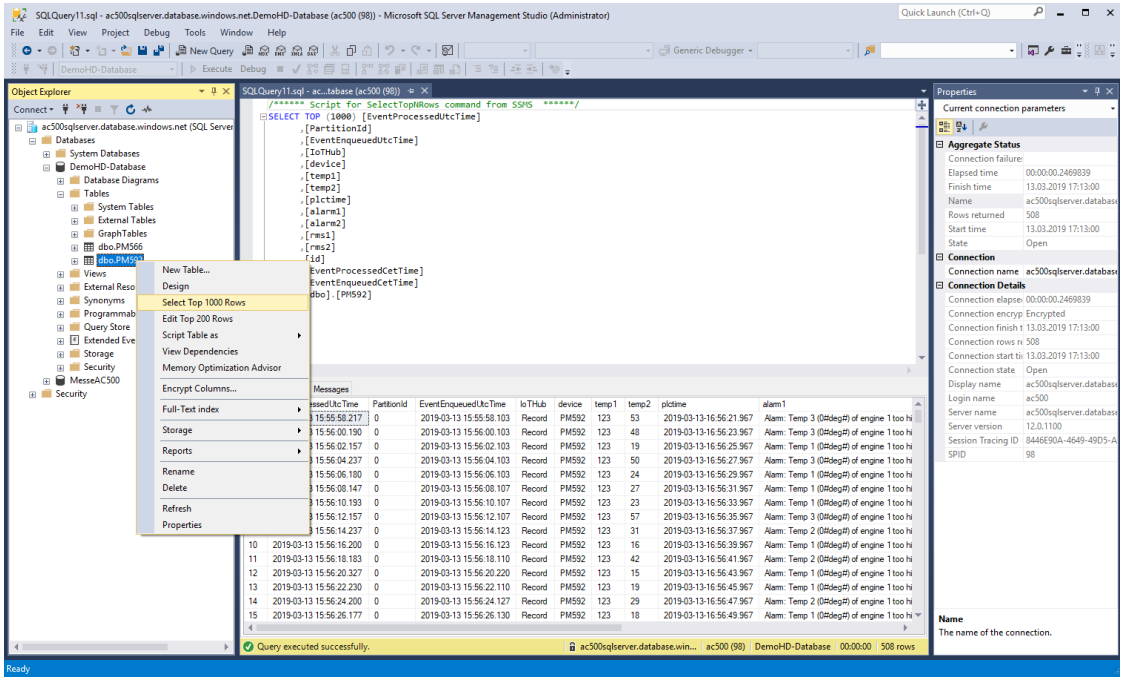
The table structure is the following:

Column Name	Data Type	Allows Nulls
EventProcessedUtcTime	datetime	NULL
PartitionId	int	NULL
EventEnqueuedUtcTime	datetime	NULL
IoTHub	text	NULL
device	text	NULL
temp1	real	NULL
temp2	real	NULL
plctime	text	NULL
alarm1	text	NULL
alarm2	text	NULL
rms1	real	NULL
rms2	real	NULL
id	int	IDENTITY(1,1) NOT NULL
EventProcessedCetTime	datetime	NULL
EventEnqueuedCetTime	datetime	NULL

id has PRIMARY KEY

The result will be the creation of the table “PM592”.

Right click on the table PM592 and choose: “Select Top 1000 Rows”. The result will be the following screen, with the data inside, if the PLC is already running.



3.3 Configuration of MS Azure services

After creation of Azure services and database, the configuration can be done.

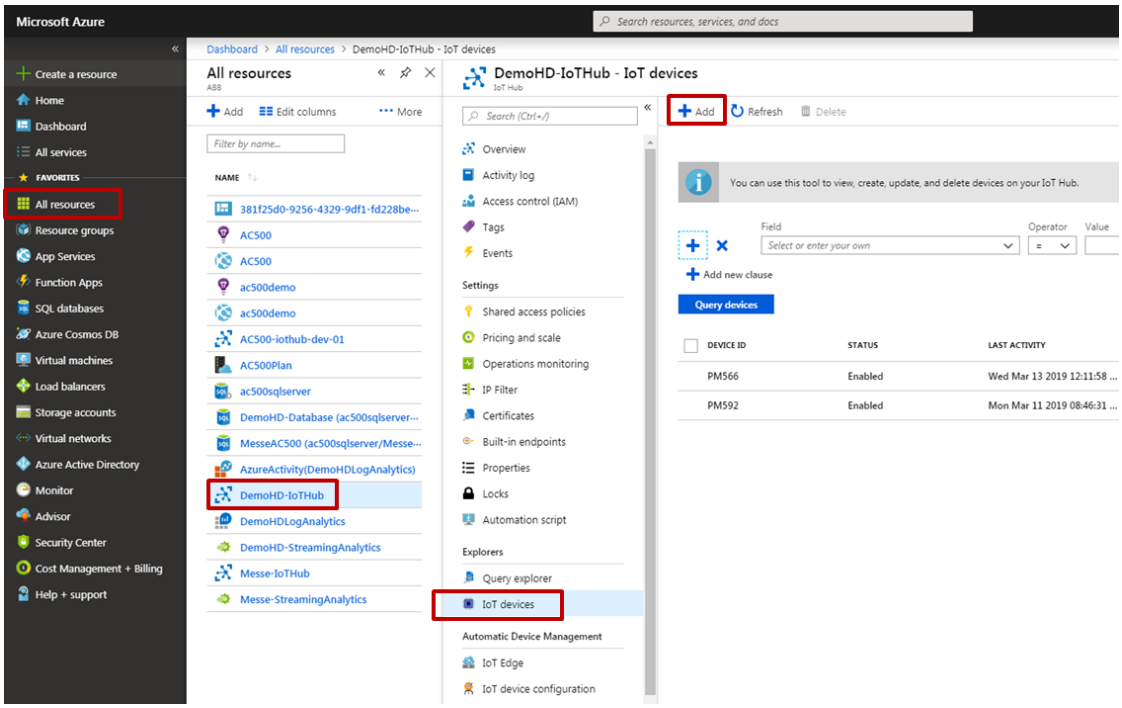
3.3.1 IoT Hub & device configuration

To enable communication between local devices (e.g. PM592) and the cloud, they need to be registered in the IoT Hub and a shared signature containing username and password for authentication has to be created.

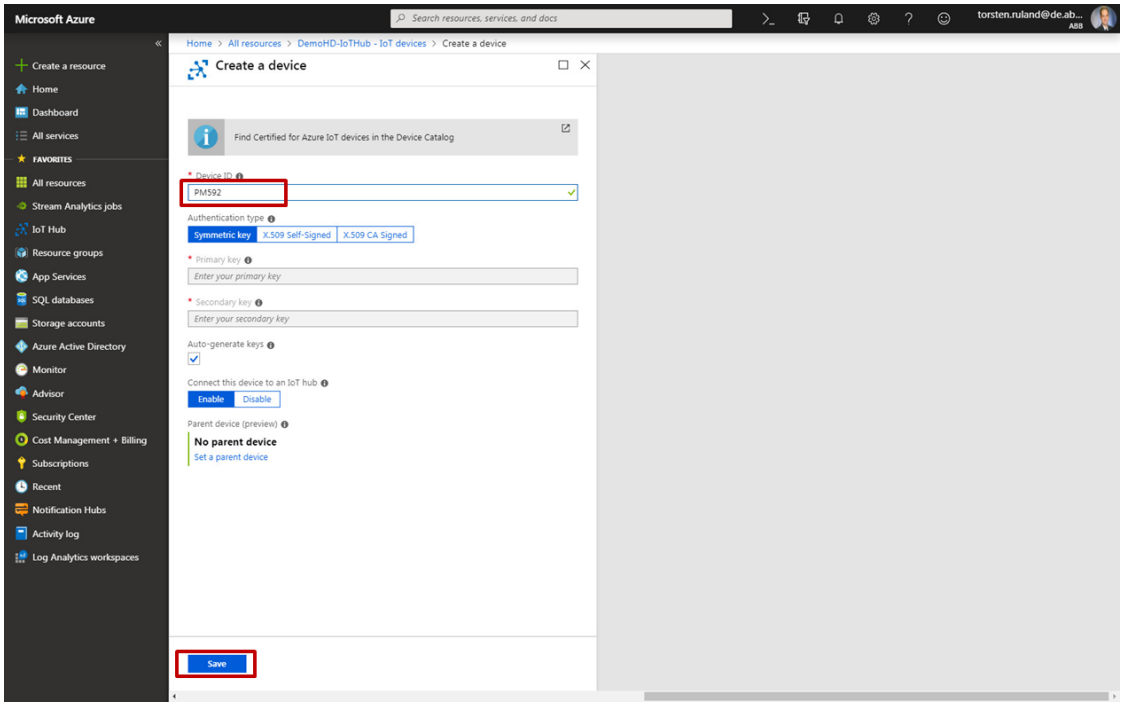
This is how to proceed:

1) Add devices to the IoT Hub

- Go to **All resources**
- Select the **IoT Hub**
- Select **IoT devices**
- Click **Add**



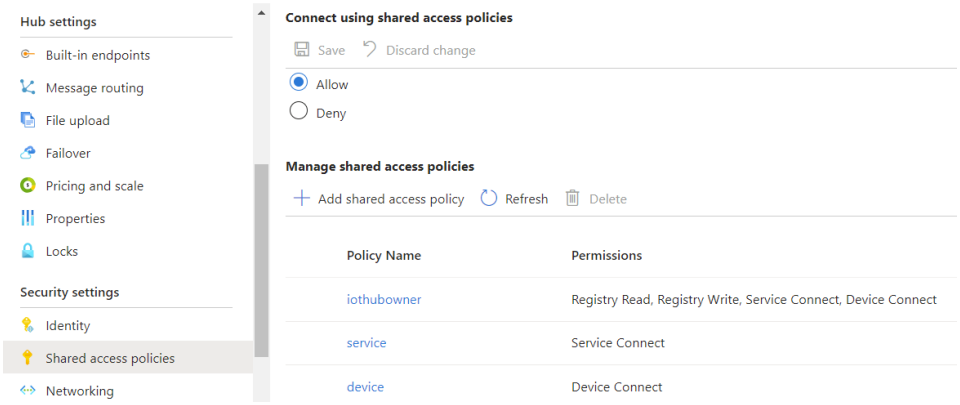
- Name the device (Device Id)
  - use **unique Ids**
  - do **not** use **underscores _**
- Click **Save**



2) Generate Username + Password with Azure IoT Explorer

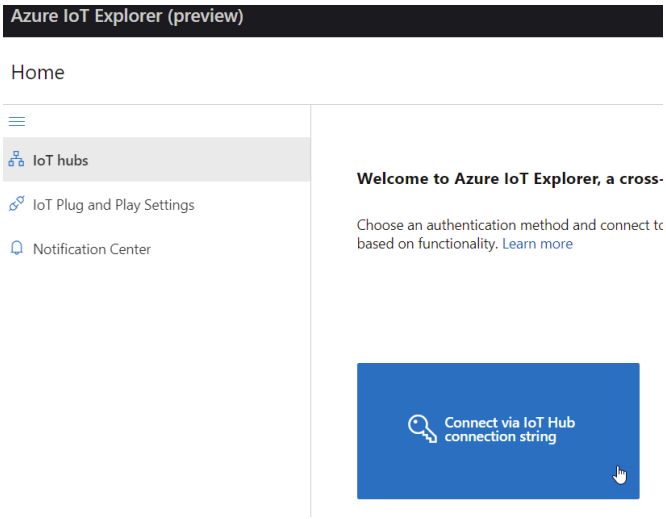
For authentication of local devices, username and password must be configured using MS Azure credentials and the Azure IoT Explorer.

- Open the **IoT Hub** in **Microsoft Azure Portal**
- Select **Security settings** > **Shared access policies**

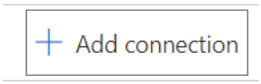


- Click **iothubowner** and copy the **Primary connection string**

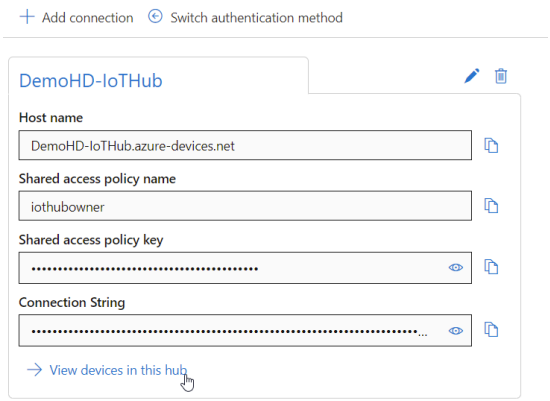
- Download and install / open Azure IoT Explorer or any similar software
- Select **IoT hubs** and click "**Connect via IoT Hub connection string**"



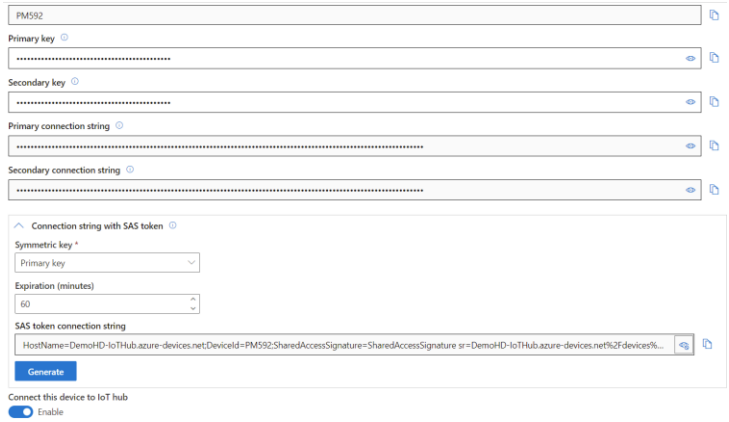
- Click **Add connection** and parse the copied primary connection string



- Click **View devices in this hub**



- Select the required device (Here "PM592")
- Expand **Connection string with SAS token**



- **Select Primary** key as **Symmetric key**

- Define the **Expiration** time for the Shared Access signature in minutes. The table below gives some standard times in minutes:

Time	1 hour	1 day	1 week	1 month	1 year	5 years	10 years
In minutes	60	1440	10080	43200	525600	2628000	525600

Note: Once the validation time is expired the device is no longer able to connect to Azure IoT hub.

- Click **Generate** to generate a SAS token. This is looking like shown below:

  HostName=DemoHD-IoTHub.azure-devices.net;DeviceId=PM592;
  SharedAccessSignature=SharedAccessSignature sr=DemoHD-IoTHub.azure-devices.net%2Fdevices%2FPM592&sig=0sqfp08OMiEfuezLqjv%2FcM3GsHgZt3pZGFfR7JFN%2B%2B0%3D&se=17060xxxxx

  In the PLC programming, Hostname, ClientID, Username and Password are required. These can be found inside the SAS token:

Hostname	DemoHD-IoTHub.azure-devices.net
ClientID	PM592
Username	DemoHD-IoTHub.azure-devices.net/PM592
Password	SharedAccessSignature sr=DemoHD-IoTHub.azure- devices.net%2Fdevices%2FPM592&sig=0sqfp08OMiEfuezLqjv %2FcM3GsHgZt3pZGFfR7JFN%2B%2B0%3D&se=17060xxxxx

Following IEC code snippet can be used to get Hostname, ClientID, Username and Password from the SAS token.

```
(*Maximum SAS length: 254 chars. If longer String Utils library must be used*)
FUNCTION AzureSASParser : BOOL
VAR_INPUT
	SasTokenConnectionString: STRING(255) := 'HostName=<AzureHostName>;De-
viceId=<AzureClientId>;SharedAccessSignature=<SASContainingDevicePathAndSignature>';
END_VAR
VAR_OUTPUT
	sHostname: STRING(255);
	sClientID: STRING(255);
	sUsername: STRING(255);
	sPassword: STRING(255);
END_VAR
VAR
	iFindPos: INT := 0;
	iStartPos: INT;
END_VAR
```

```
(* Hostname*)
iFindPos := iFindPos := FIND(SasTokenConnectionString, 'HostName=');
IF iFindPos = 0 THEN RETURN; END_IF
iStartPos := iFindPos + 9; (* HostName= are 9 chars*)
iFindPos := FIND(SasTokenConnectionString, ';DeviceId');
IF iFindPos = 0 THEN RETURN; END_IF
sHostname := MID(SasTokenConnectionString, iFindPos-iStartPos, iStartPos);

(* DeviceId*)
iFindPos := FIND(SasTokenConnectionString, 'DeviceId=');
IF iFindPos = 0 THEN RETURN; END_IF
iStartPos := iFindPos + 9; (* DeviceId= are 9 chars*)
iFindPos := FIND(SasTokenConnectionString, ';Shared');
IF iFindPos = 0 THEN RETURN; END_IF
sClientID := MID(SasTokenConnectionString, iFindPos-iStartPos, iStartPos);

(* SharedAccessSignature*)
iFindPos := FIND(SasTokenConnectionString, 'SharedAccessSignature=');
IF iFindPos = 0 THEN RETURN; END_IF
iStartPos := iFindPos + 22; (* SharedAccessSignature= are 22 chars*)
iFindPos := LEN(SasTokenConnectionString)+1; (* Last entry --> Stringlength *)
IF iFindPos = 0 THEN RETURN; END_IF
sPassword := MID(SasTokenConnectionString, iFindPos-iStartPos, iStartPos);

(* Username *)
sUsername := CONCAT(sHostname, '/');
sUsername := CONCAT(sUsername, sClientID);

AzureSASParser := TRUE;
```

3.3.2 Stream Analytics Configuration

To process the content of messages sent to Azure, the Stream Analytics service is required. Using SQL-like queries, specific parameters from the message can be selected and modified and forwarded to the desired outputs (e.g. a SQL-database).

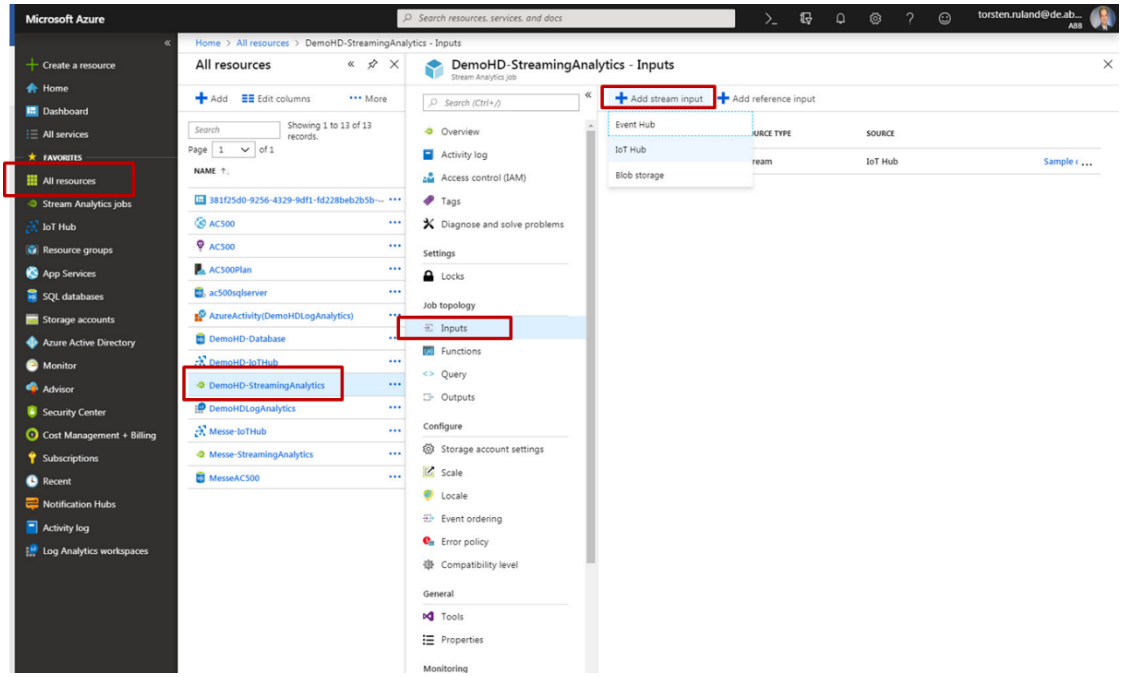
In Stream Analytics, inputs & outputs must be created.

The input is indicating the source of information (here: the IoT Hub, where the messages arrive).

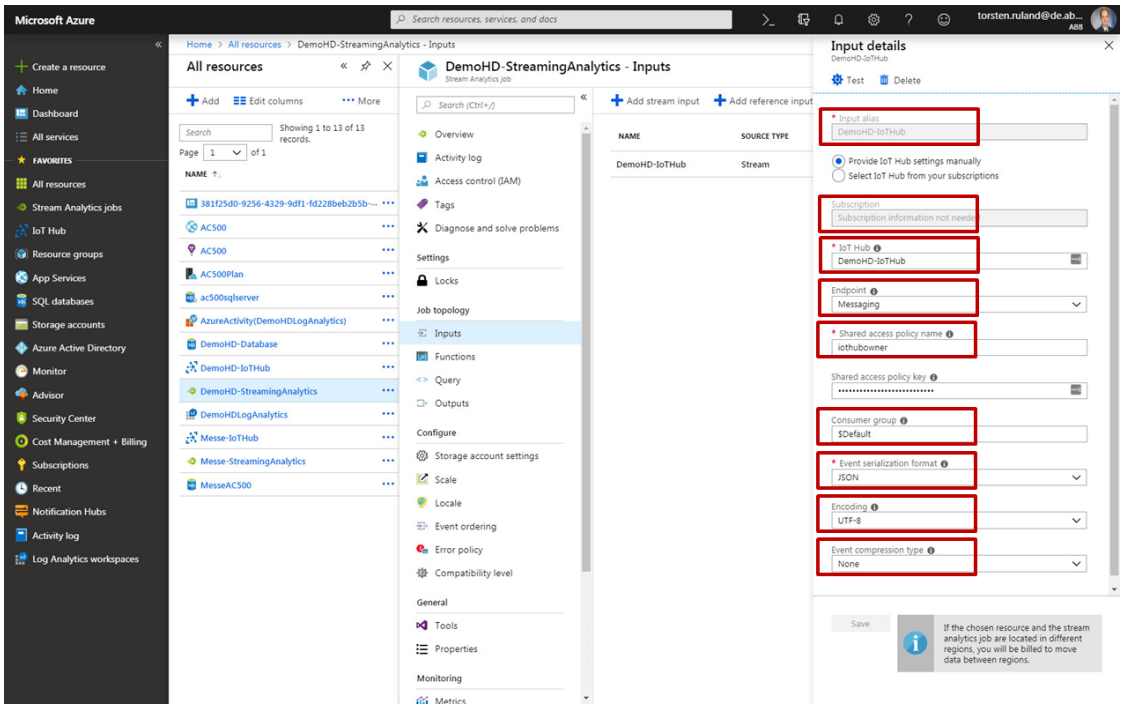
The output is indicating, where the data should be sent to (here: SQL-Database for data storage and Power BI for real-time monitoring & visualization).

1) Create an IoT Hub input in Azure Stream Analytics

- Go to **All resources**
- Select the **Stream Analytics Job**
- Select **Inputs**
- Select **Add stream input**
- Select **IoT Hub**

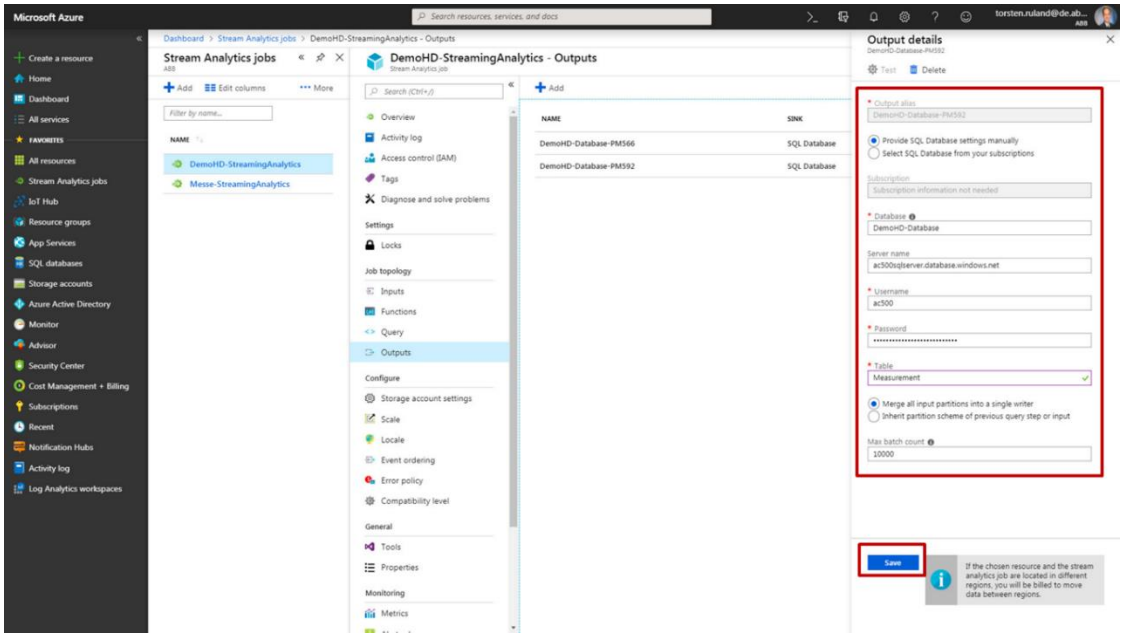


- Enter the following information:
  - Input alias: *Name the input (any name)*
  - Subscription: *Choose the subscription*
  - IoT Hub: *Choose the IoT Hub created before*
  - Endpoint: **Messaging**
  - Shared access policy name: **iothubowner**
  - Consumer group: **$Default**
  - Event serial. Format: **JSON**
  - Encoding: **UTF-8**
  - Event compression type: **None**



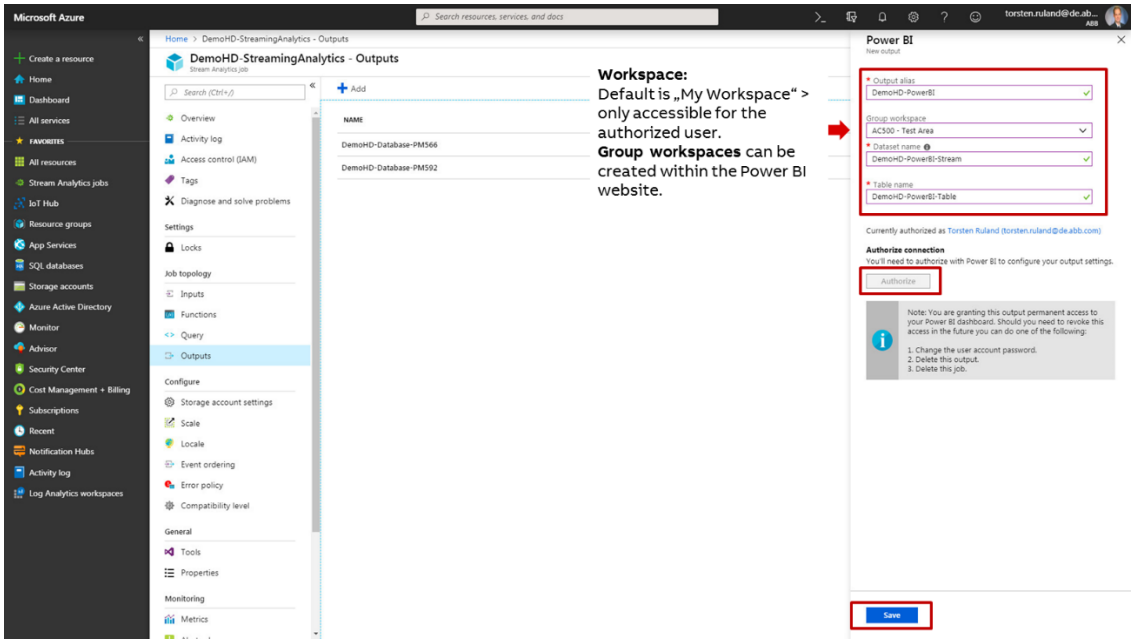
2) Create a SQL-Database output in Azure Stream Analytics

- Select **Outputs**
- Select **Add**
- Select **SQL-Database**
- Enter the following information
  - Output alias: *Name the output (any name)*
  - Subscription: *Choose the subscription*
  - Database: *Choose the SQL-Database created before*
  - Server name: *Choose the SQL-server created before*
  - Username: *Enter "**Server admin login**" of SQL-server created in* Creation of MS Azure services *3.1*
  - Password: *Enter the "**Password**" of SQL-server created in* Creation of MS Azure services *3.1*
  - Table: *Name the table for storing the data (table must exist in the database)*
- Press **Save**



3) Create a Power BI output in Azure Stream Analytics

- Select **Outputs**
- Select **Add**
- Select **Power BI**
- Enter the following information:
  - Output alias: *Name the output (any name)*
  - Group Workspace: *Define to which workspace data should be sent*
    - Default is "**My workspace**":
      - only accessible for the authorized user
      - can be used without Power BI license
    - **Group workspaces**
      - Enable sharing dashboards & reports with several users
      - Can only be used with Power BI Pro / Premium license
    - Additional information on workspaces
  - Dataset name: *Name the dataset in Power BI*
  - Table name: *Name the table in Power BI*
- Press **Authorize**
- Enter **e-mail-address** of authorized user
- Press **Save**



4) Program query

After having created all the inputs & outputs, the query can be written. **Stream Analytics queries** are written with **SQL-like syntax**.

The basic elements of the query are the following:

Query element	Explanation
SELECT	Defines which values should be selected from the incoming data stream. Default is “ * ” which means that all data is taken and forwarded. **Note 1:** for writing data into the SQL-Database, a specific selection of parameters is recommended as precise names & IDs for each column inside the database-table must be defined. As the IoT Hub is adding parameters to the messages, there could arise issues otherwise. Also, a modification of parameters is possible. (e.g. adding hours to the timestamps given by the IoT Hub) **Note 2:** the order of selection is relevant as well. Therefore, please check with the structure of the message. Data that has been added by Azure IoT Hub will be on top of the message.
INTO	Defines to which **outputs** selected data should be forwarded.
FROM	Defines from which **inputs** data should be processed.

For more complex queries, there are **additional elements** available.

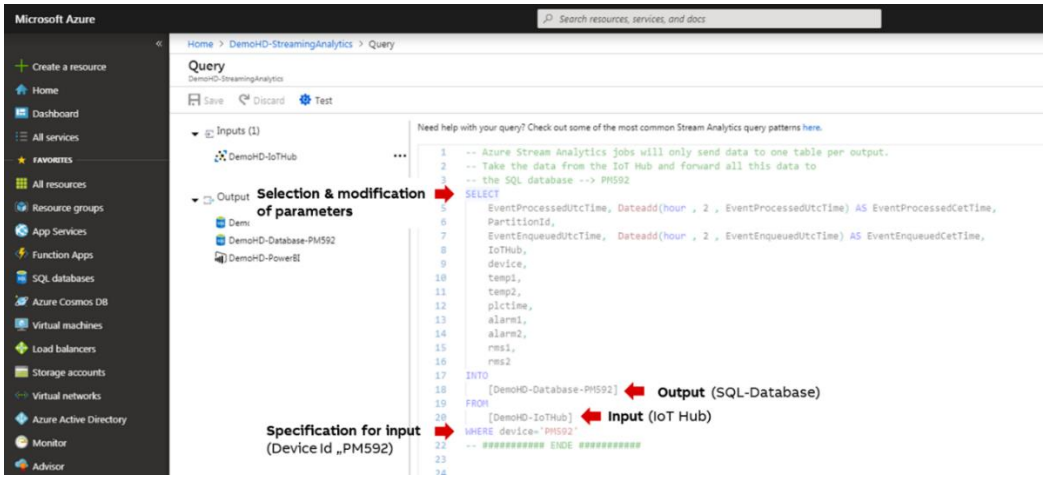
Please check MS Azure for additional information.

Stream Analytics query in demo application

For every output, a separate query must be used.

In the demo application, the query looks like this:

Query 1 for SQL-Database output



- **SELECT**
  - Takes the above-mentioned parameters from the message
  - Uses function **Dateadd (+ 2 hrs)** to change timestamp from UTC to CET-time
- **INTO**
  - Defines **SQL-Database** as output.
- **FROM**
  - Defines **IoT-Hub** as input.
- **WHERE**
  - Specifies that only data from **device** "**PM592**" should be taken from input.
    This "device" string is coming from the PLC JSON payload. In this example it is required that each message has the member "device":"PM592".

    For further information check the application examples AC500 MQTT library and AC500 V3 JSON library.

Query 2 for Power BI output

- **SELECT**
  - Takes the above-mentioned parameters from the message
  - Uses function **Dateadd (+ 2 hrs)** to change timestamp from UTC to CET-time
- **INTO**
  - Defines **Power BI** as output.
- **FROM**
  - Defines **IoT-Hub** as input.

```
48  -- Azure Stream Analytics jobs will only send data to one table per output.
49  -- Take the data from the IoT Hub and forward all this data to
50  -- the PowerBI
51  SELECT
52      EventProcessedUtcTime, Dateadd(hour , 2 , EventProcessedUtcTime) AS EventProcessedCetTime,
53      PartitionId,
54      EventEnqueuedUtcTime,  Dateadd(hour , 2 , EventEnqueuedUtcTime) AS EventEnqueuedCetTime,
55      IoTHub,
56      device,
57      temp1,
58      temp2,
59      plctime,
60      alarm1,
61      alarm2,
62      rms1,
63      rms2
64  INTO
65      [DemoHD-PowerBI]
66  FROM
67      [DemoHD-IoTHub]
68  -- ########## ENDE ###########
```

3.3.3 Service App Configuration

This part will not be covered by the documentation.
Please refer to the MS Azure documentation for more information.

3.3.4 Power BI Configuration

The easiest and fastest way to achieve cloud visualization is done by using Power BI, a MS product for Business Intelligence. It enables creation of dashboards, reports and other elements.

It can be used for free under the following conditions:

- A MS account (e.g. through corporate e-mail-address) is required
- You are the authorized user mentioned in 3.3.2 and use “My workspace”

(It means that only you have access to the data and nobody else can see reports created in Power BI)

The free version, however, only offers limited functionality (e.g. creation of reports).

With **the Power BI Pro** or **Power BI Premium license**, the **full functionality** can be used to share data, dashboards, reports, etc. with others.

To find out more about Power BI licenses, please check Power BI.

Creation of dashboards

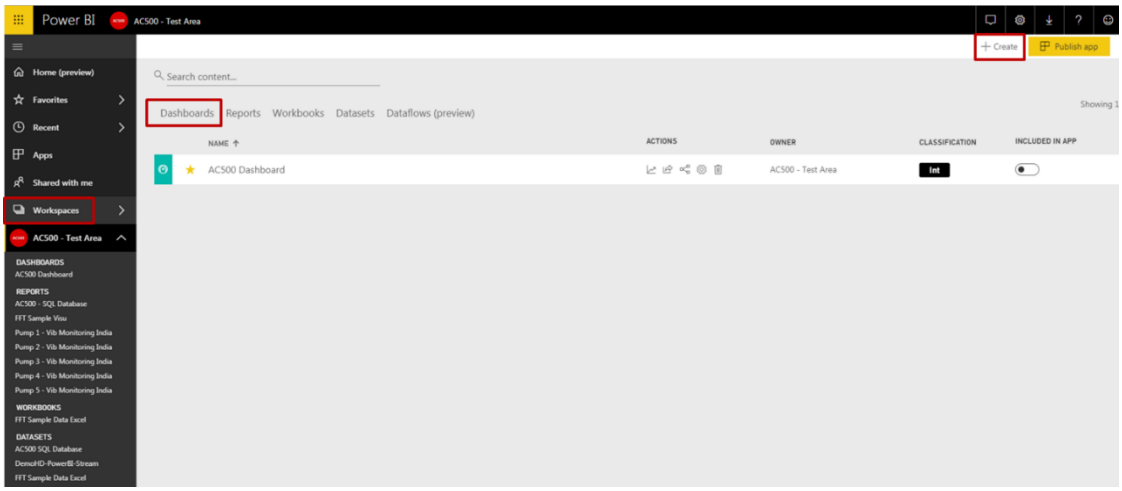
This documentation will describe how a dashboard can be created in Power BI.
A dashboard is a canvas where graphics & data from streaming datasets (e.g. the dataset created in 3.3.2) can be implemented. Dashboards are temporary and update automatically – as long as the chosen data stream is sending data.

For the creation of dashboards, a **Power BI license** is **required**.

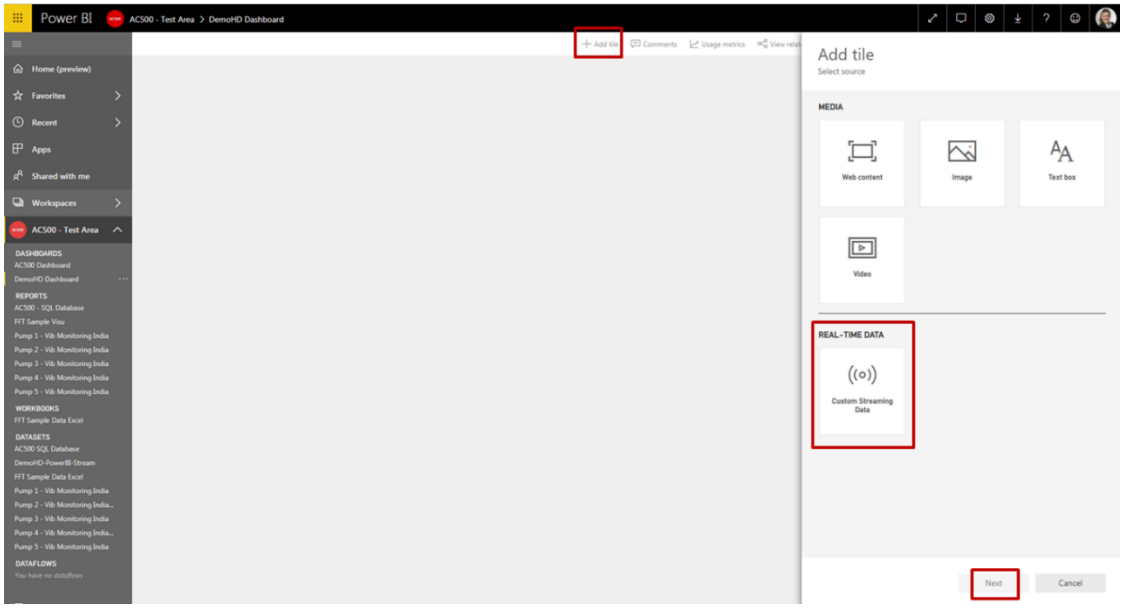
Please check the MS documentation for further information on dashboards.

To create a dashboard, proceed like this:

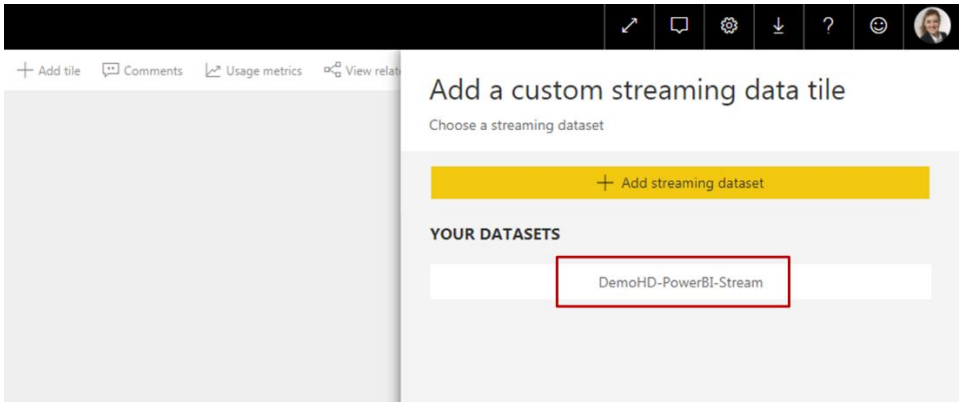
- Log in to Power BI
- Press **Workspaces**
- Select the Workspace chosen in 3.3.2
  (Here the group workspace: “AC500 – Test Area” is choosen)
- Press **Dashboards**
- Press **Create**



- Press **Add tile**
- Select **Custom Streaming Dataset**
- Press **Next**

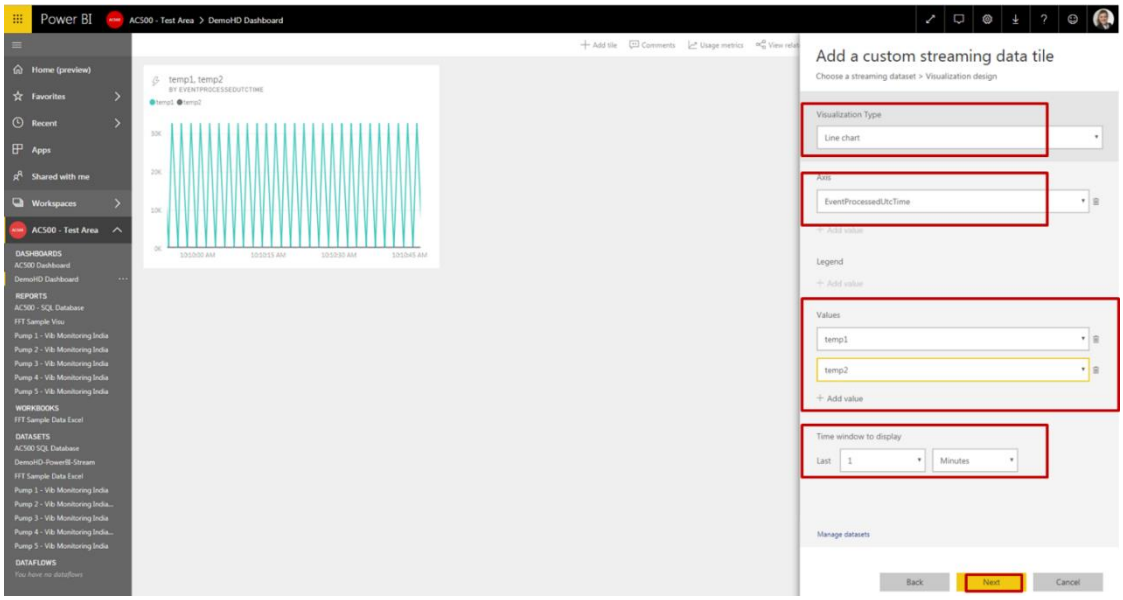


- Select the dataset created in 3.3.2
  Here: “Demo-HD-PowerBI-Stream”



Enter following information

- Visualization Type — **Line Chart**
- Axis: — *Choose the timestamp for the x-axis*
  Here: EventProcessedUtcTime
- Values: — *Choose the parameters that should be show*
  Here: temp1, temp2
- Time window to display: — *Choose the desired time window*
  Here: Last 1 min



- Press **Next**
- *Name the tile*
- **Optional**: set a link to
  - An external website
  - Another Power BI document in the chosen workspace (e.g. report or another dashboard)
- Press **Apply**

Now a dashboard for real-time monitoring of the temperature values is created.
More features (e.g. alerts) can be added to this dashboard.

To find out more, please check the Power BI documentation.

4 PLC Configuration

The settings how to connect from an AC500 PLC to an MQTT broker are explained in the application example AC500 MQTT library. Details about the JSON format and how to use this in the AC500 V3 PLC are explained in the application example AC500 V3 JSON library.

4.1 Publish and subscribe topics

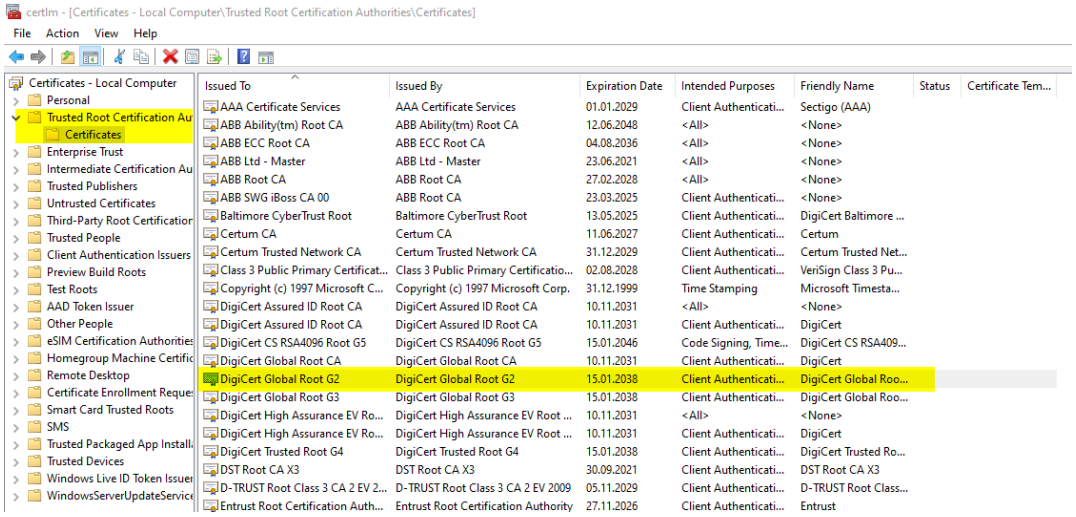
In this MS Azure example the topics to send and subscribe data are:

- devices/{device_id}/messages/events/ - (here: devices/PM592/messages/events/)
- devices/{device_id}/messages/devicebound/# - (not used in this example)

4.2 Certificate for Azure

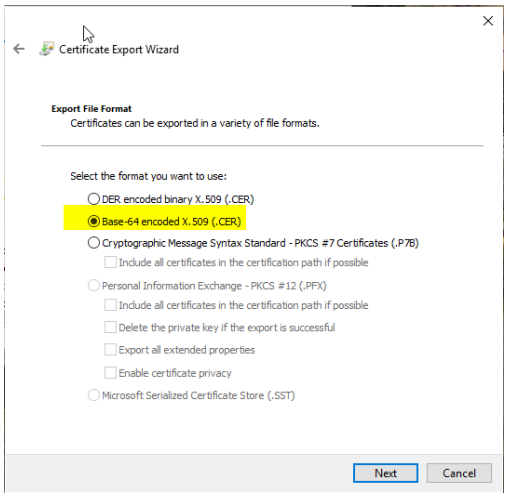
4.2.1 DigiCert

Azure IoT Hub is using the “DigiCert Global Root G2” root certificate for signing. This can be extracted from Windows CertMgr. (Run “certmgr.msc”)



Right click → All Tasks → Export

Please be sure, that you extract the file as “Base-64” encoded:



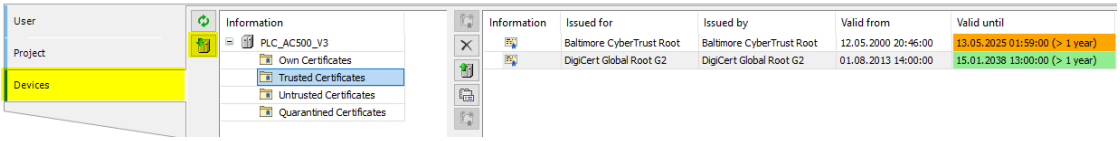
4.2.2 How to change from Baltimore to DigiCert?

“Starting in February 2023, all IoT hubs in the global Azure cloud will migrate to a new TLS certificate issued by the DigiCert Global Root G2.” https://learn.microsoft.com/en-us/azure/iot-hub/migrate-tls-certificate?tabs=portal

To change the root certificate steps in the AC500 and the Azure portal are necessary.

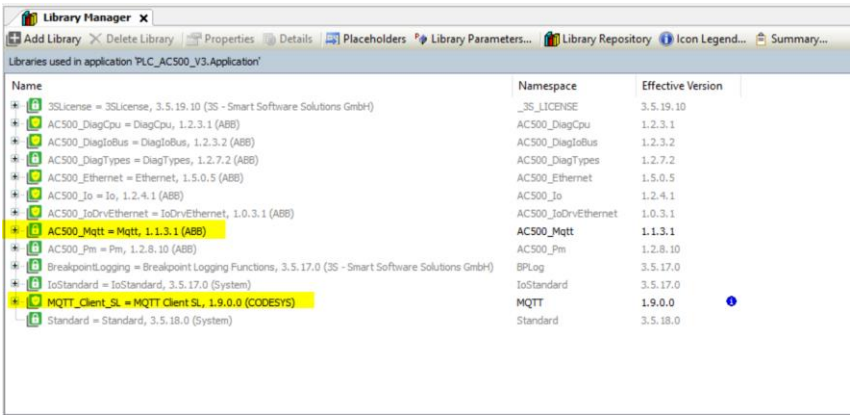
Attention not only the root certificate but also the IP address is changing. A DNS resolve inside the program is required. If missing, please add before trying to connect.

1. Export the **DigiCert Global Root G2 as described in chapter** Error! Reference source not found.
2. Connect to the PLC and open the Security Screen
3. Open Devices, select Trusted Certificates and install the new certificate

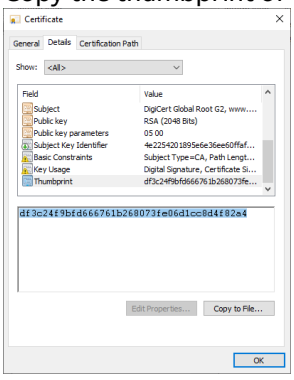


4. If using “AC500_MQTT” library, continue here.
   If using “MQTT Client SL” library, continue with step 8.

   Please use only one of both libraries.



5. Copy the thumbprint of the certificate



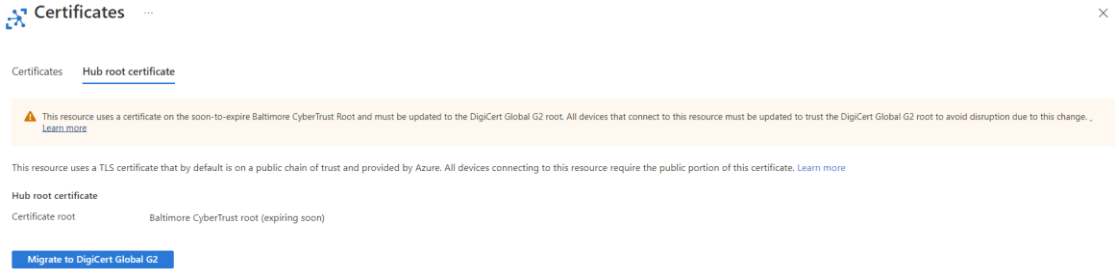
6. In the code, where the connect fb is called, change the ServerCert

```
10: //Connect
    Connect(Execute := TRUE,
            Conn := ADR(MQTT.Connection),
            IpAddress := DnsResolve.IPAdrString,
            Port := 8883,
            Secure := TRUE,
            ServerCert := 'df3c24f9bfd666761b268073fe06d1cc8d4f82a4',
```

7. Login to the PLC via online change and create a boot project

The existing MQTT connection will remain only new connections will be done via the new certificate.

8. Login to azure portal
9. Open the IoT hub and click “Migrate to DigiCert Global G2”



10. Check all boxes and click Update

Please be patient. The update might take some time

11. After the update check if the PLC is still connected. Trigger a new connect to check if the connection with the new certificate is working properly.

5 FAQs

5.1 Which QoS does Azure support?

Please check:

https://docs.microsoft.com/de-de/azure/iot-hub/iot-hub-mqtt-support

Azure IoT Hub does **not** support QoS 2 messages.

When a device app publishes a message with QoS 2, IoT Hub closes the network connection.

5.2 Does Azure support multiple connections with the same device?

Please check:

https://docs.microsoft.com/de-de/azure/iot-hub/iot-hub-mqtt-support

Azure IoT Hub only supports one active MQTT connection per device. Each new MQTT connection for the same device ID causes the IoT Hub to disconnect the existing connection.

5.3 Does Azure support Retain flag?

Please have a look at this link:

https://docs.microsoft.com/de-de/azure/iot-hub/iot-hub-mqtt-support

IoT Hub does not persist retention messages. When a device sends a message with the RETAIN flag set to 1, IoT Hub adds the application property x-opt-retain to the message. In this case, IoT Hub does not persist the retention message, but passes it on to the back-end app.

5.4 What are the costs for sending and storing data on Azure?

It depends on the amount of data that will be send. Each data they will be send to Azure, will be received by a generic broker, called IoT Hub on Azure side. This data will be stored in the broker database until the Streaming Analytics job from Azure access the data and forward them, for example to a SQL database. This means, as more data they will be send to Azure, as more you must pay.

ABB AG

Contact:
https://access.motion.abb.com/contact/contact

Homepage:
www.abb.com/plc