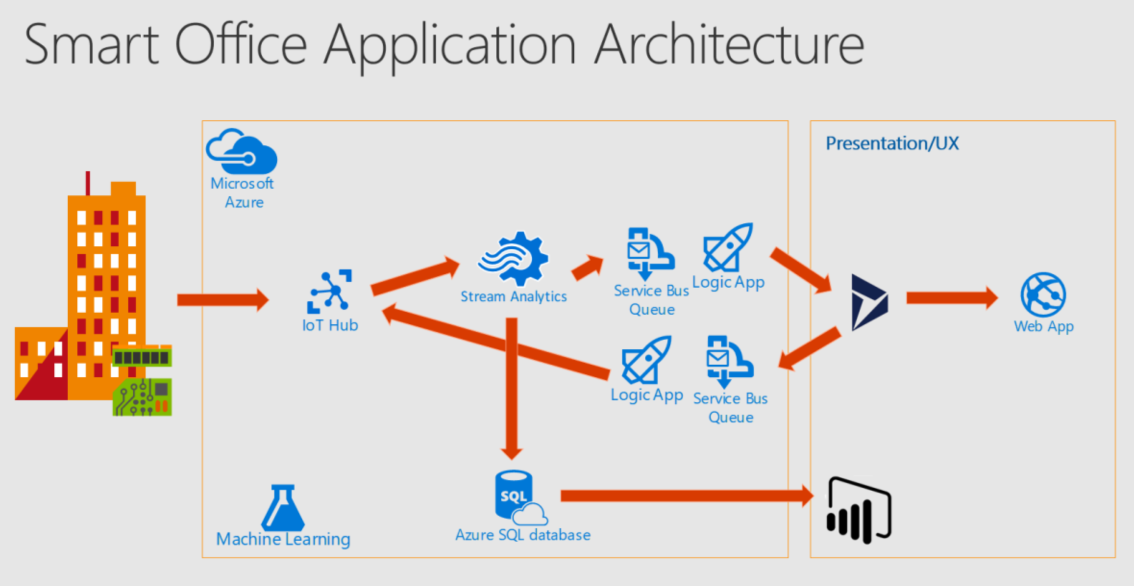
SMART OFFICE APPLICATION

Pedro Sacramento

MICROSOFT

Contents

Smart Office Application Solution Architecture



Microsoft Dynamics 365 for Customer Engagement

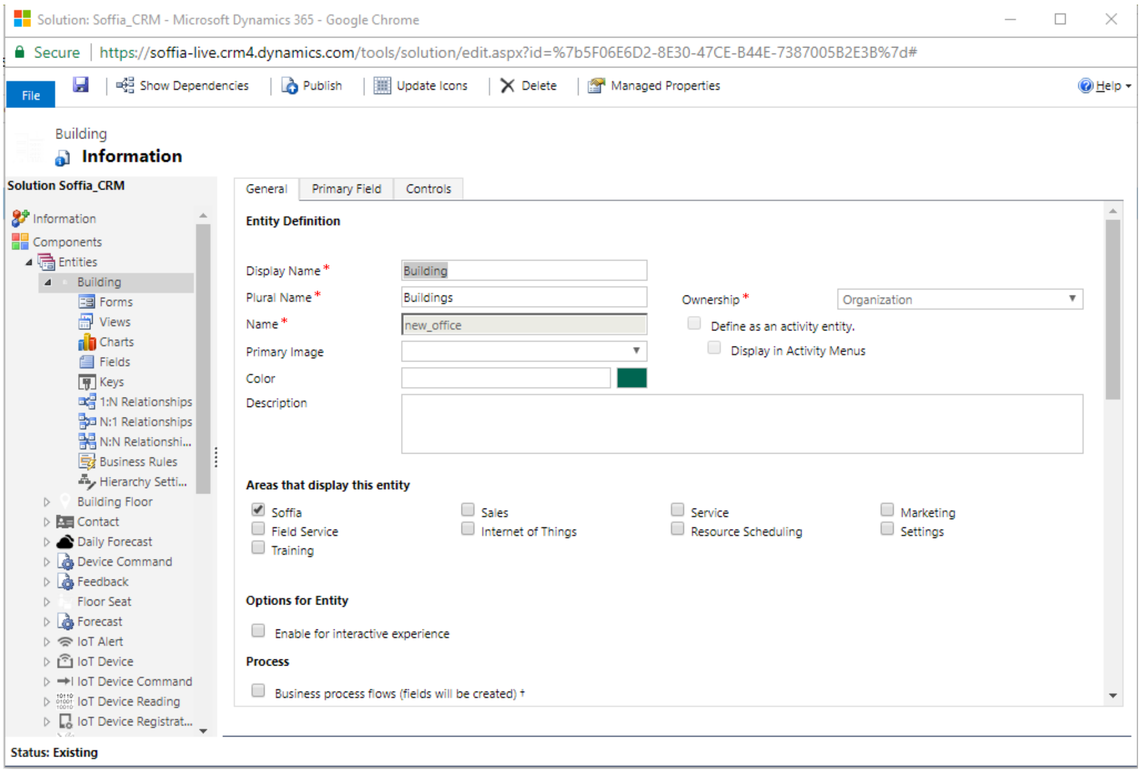
Solutions

The first step in Dynamics 365 for Customer engagement is to create the customizations for the Smart Office Application scenario.

To help with this process we've shared the Soffia_CRM solution.

If you would like to create your own logic or build your own solution, here's a brief explanation of the entities that were used for this scenario.

The main entity is "Building", you'll find references to the name "Office" but to avoid confusions with Microsoft Office we've changed the display name to "Building".



Entity Definition

Display Name * Building

Plural Name * Buildings

Name * new_office

Primary Image

Color

Description

Ownership * Organization

□ Define as an activity entity.

□ Display in Activity Menus

Areas that display this entity

☑ Soffia □ Sales □ Service □ Marketing

□ Field Service □ Internet of Things □ Resource Scheduling □ Settings

□ Training

Options for Entity

□ Enable for interactive experience

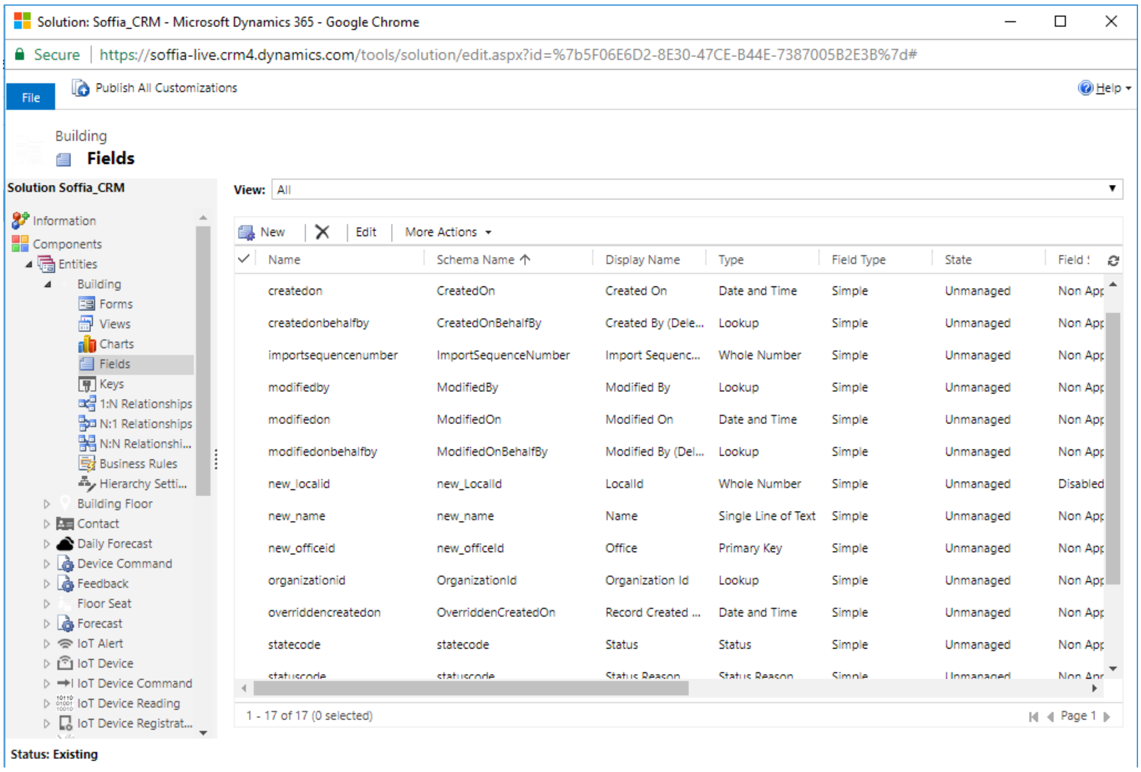
Process

□ Business process flows (fields will be created) †

Status: Existing

We've also customized the sitemap and added a new Menu named "Soffia". There are multiple approaches to achieve this, in the latest versions of Dynamics 365 for CE you have available a native site map editor.

The field local id stores the value needed to display weather information for the building location.



Solution: Soffia_CRM - Microsoft Dynamics 365 - Google Chrome

Secure | https://soffia-live.crm4.dynamics.com/tools/solution/edit.aspx?id=%7b5F06E6D2-8E30-47CE-B44E-7387005B2E3B%7d#

Building

Fields

Solution Soffia_CRM

View: All

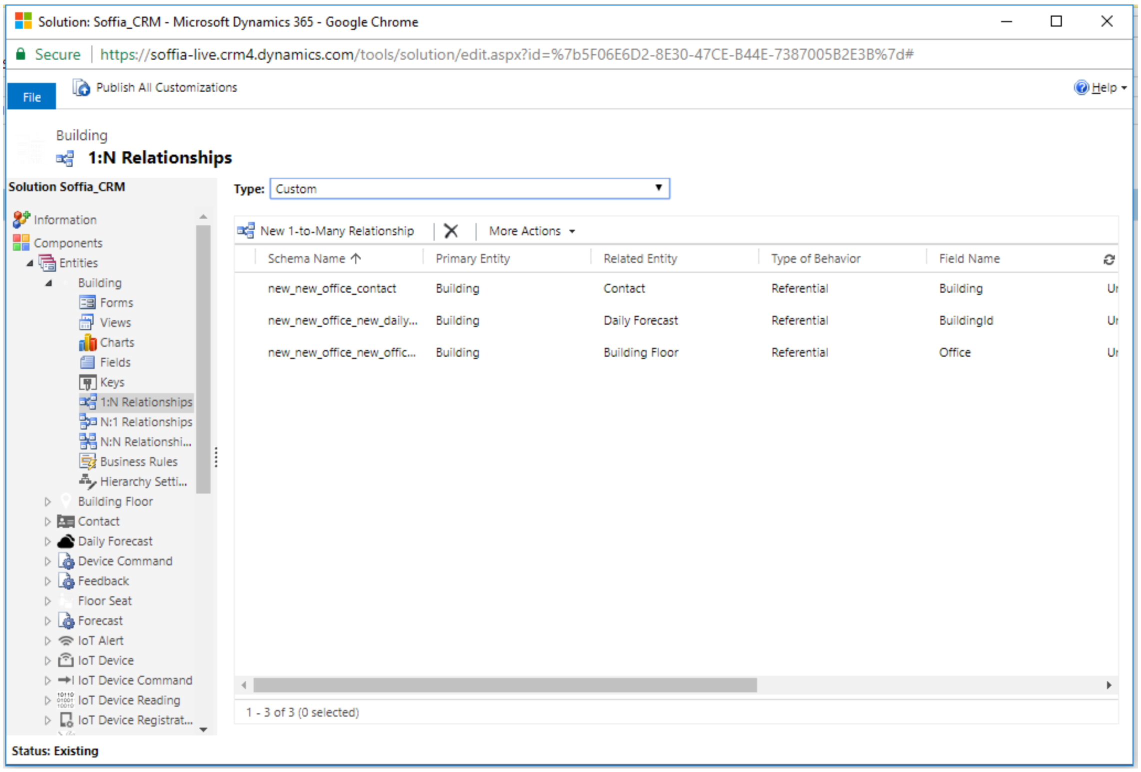
Name	Schema Name	Display Name	Type	Field Type	State	Field !
createdon	CreatedOn	Created On	Date and Time	Simple	Unmanaged	Non App
createdonbehalfby	CreatedOnBehalfBy	Created By (Dele...	Lookup	Simple	Unmanaged	Non App
importsequencenumber	ImportSequenceNumber	Import Sequenc...	Whole Number	Simple	Unmanaged	Non App
modifiedby	ModifiedBy	Modified By	Lookup	Simple	Unmanaged	Non App
modifiedon	ModifiedOn	Modified On	Date and Time	Simple	Unmanaged	Non App
modifiedonbehalfby	ModifiedOnBehalfBy	Modified By (Del...	Lookup	Simple	Unmanaged	Non App
new_localid	new_LocalId	LocalId	Whole Number	Simple	Unmanaged	Disabled
new_name	new_name	Name	Single Line of Text	Simple	Unmanaged	Non App
new_officeid	new_officeId	Office	Primary Key	Simple	Unmanaged	Non App
organizationid	OrganizationId	Organization Id	Lookup	Simple	Unmanaged	Non App
overriddencreatedon	OverriddenCreatedOn	Record Created ...	Date and Time	Simple	Unmanaged	Non App
statecode	statecode	Status	Status	Simple	Unmanaged	Non App
statuscode	statuscode	Status Reason	Status Reason	Simple	Unmanaged	Non App

1 - 17 of 17 (0 selected)

Page 1

Status: Existing

There is a 1:N relationships between "Building" entity and "Building Floor".



Solution: Soffia_CRM - Microsoft Dynamics 365 - Google Chrome

Secure | https://soffia-live.crm4.dynamics.com/tools/solution/edit.aspx?id=%7b5F06E6D2-8E30-47CE-B44E-7387005B2E3B%7d#

Building

1:N Relationships

Solution Soffia_CRM

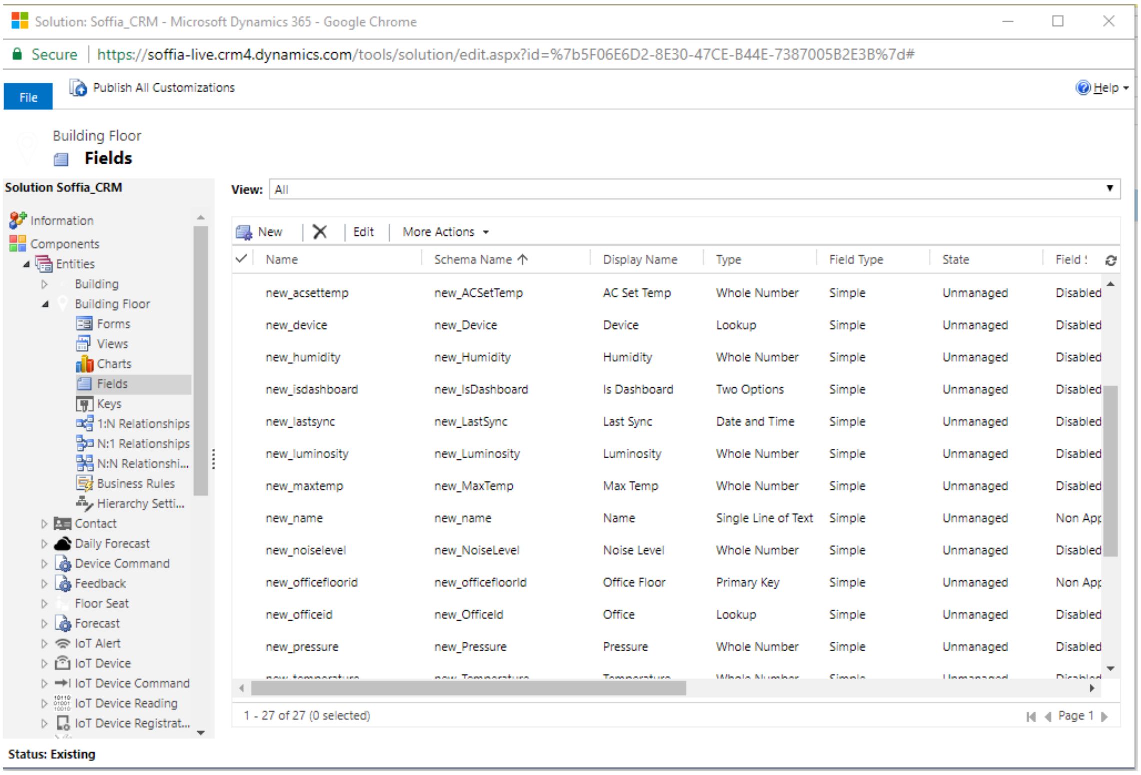
Type: Custom

Schema Name	Primary Entity	Related Entity	Type of Behavior	Field Name
new_new_office_contact	Building	Contact	Referential	Building
new_new_office_new_daily...	Building	Daily Forecast	Referential	BuildingId
new_new_office_new_offic...	Building	Building Floor	Referential	Office

1 - 3 of 3 (0 selected)

Status: Existing

The entity "Building Floor" will contain the wings in each floor that are part of the building and will store the information that is collected from the devices distributed across the site.



Name	Schema Name	Display Name	Type	Field Type	State	Field !
new_acsettemp	new_ACSetTemp	AC Set Temp	Whole Number	Simple	Unmanaged	Disabled
new_device	new_Device	Device	Lookup	Simple	Unmanaged	Disabled
new_humidity	new_Humidity	Humidity	Whole Number	Simple	Unmanaged	Disabled
new_isdashboard	new_IsDashboard	Is Dashboard	Two Options	Simple	Unmanaged	Disabled
new_lastsync	new_LastSync	Last Sync	Date and Time	Simple	Unmanaged	Disabled
new_luminosity	new_Luminosity	Luminosity	Whole Number	Simple	Unmanaged	Disabled
new_maxtemp	new_MaxTemp	Max Temp	Whole Number	Simple	Unmanaged	Disabled
new_name	new_name	Name	Single Line of Text	Simple	Unmanaged	Non App
new_noiselevel	new_NoiseLevel	Noise Level	Whole Number	Simple	Unmanaged	Disabled
new_officefloorid	new_officefloorId	Office Floor	Primary Key	Simple	Unmanaged	Non App
new_officeid	new_OfficeId	Office	Lookup	Simple	Unmanaged	Disabled
new_pressure	new_Pressure	Pressure	Whole Number	Simple	Unmanaged	Disabled
new_temperature	new_Temperature	Temperature	Whole Number	Simple	Unmanaged	Disabled

1 - 27 of 27 (0 selected)

Page 1

Collecting this information in Dynamics 365 for Customer Engagement depends on the intended logic.

In a common IoT scenario, information from devices can be used to trigger actions, for instance, if the temperature in a specific location is higher than the threshold, then a Work Order should be created for the issue to be fixed. There isn't really the need of displaying the actual values.

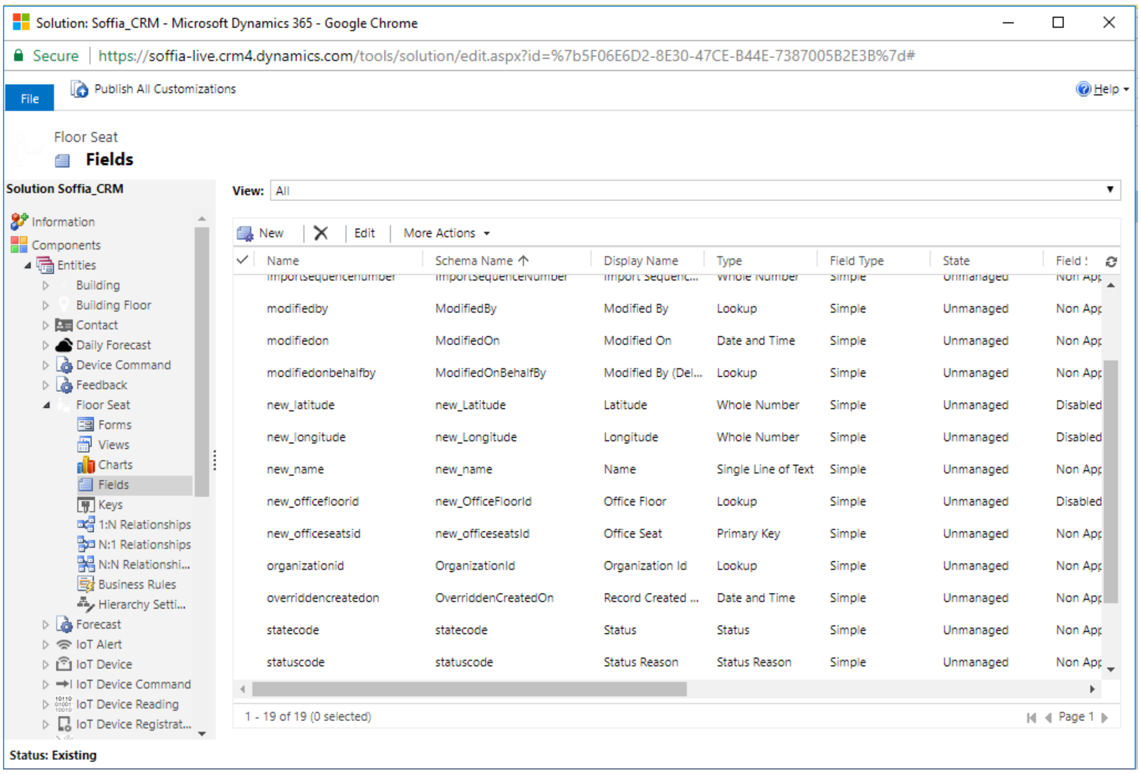
One constraint is the volume of data sent from the devices. For scenarios where there is a big number of devices and the number of messages sent is very high, it will a bad design capturing all those events in Dynamics 365 for Customer Engagement. The Azure Stack is built to receive huge amount of information from devices, trying to merge all that information into Dynamics 365 can cause issue due to the volume. The recommendation, in case, it's needed to display the information in Dynamics 365 for Customer Engagement in big volumes of data scenarios is to store the information in a SQL Azure instance, Blob storage or similar alternatives from the Azure Stack, and trigger summarized updated to the Dynamics 365 for Customer Engagement instance, reducing the amount of API calls and integration with Dynamics.

Being Dynamics 365 for Customer Engagement a Line of Business Application, causing big volumes of data to be handled by the API can cause subsequent issues and affect the user Experience with the platform.

The "Building Floor" entity will have a 1:N relationship with the entity "Floor Seat".

The entity "Floor Seat" will store all the locations where people are sitting and the location of the devices.

This will help having a blueprint of the building and understanding what areas might be affected by identified issues.



View: All

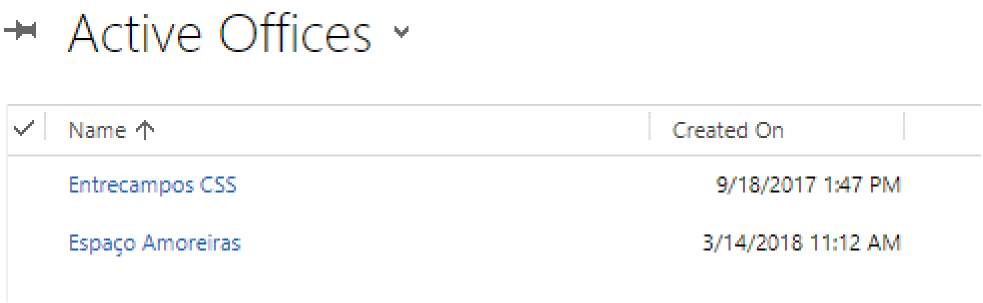
Name	Schema Name ↑	Display Name	Type	Field Type	State	Field !
importsequencenumber	ImportSequenceNumber	Import Sequenc...	Whole Number	Simple	Unmanaged	Non App
modifiedby	ModifiedBy	Modified By	Lookup	Simple	Unmanaged	Non App
modifiedon	ModifiedOn	Modified On	Date and Time	Simple	Unmanaged	Non App
modifiedonbehalfby	ModifiedOnBehalfBy	Modified By (Del...	Lookup	Simple	Unmanaged	Non App
new_latitude	new_Latitude	Latitude	Whole Number	Simple	Unmanaged	Disabled
new_longitude	new_Longitude	Longitude	Whole Number	Simple	Unmanaged	Disabled
new_name	new_name	Name	Single Line of Text	Simple	Unmanaged	Non App
new_officefloorid	new_OfficeFloorId	Office Floor	Lookup	Simple	Unmanaged	Disabled
new_officeseatsid	new_officeseatsId	Office Seat	Primary Key	Simple	Unmanaged	Non App
organizationid	OrganizationId	Organization Id	Lookup	Simple	Unmanaged	Non App
overriddencreatedon	OverriddenCreatedOn	Record Created ...	Date and Time	Simple	Unmanaged	Non App
statecode	statecode	Status	Status	Simple	Unmanaged	Non App
statuscode	statuscode	Status Reason	Status Reason	Simple	Unmanaged	Non App

1 - 19 of 19 (0 selected)   Page 1

There is an additional Entity named “Daily Forecast”, which will store all information collected from a weather API that will retrieve information for the local id of each Building.

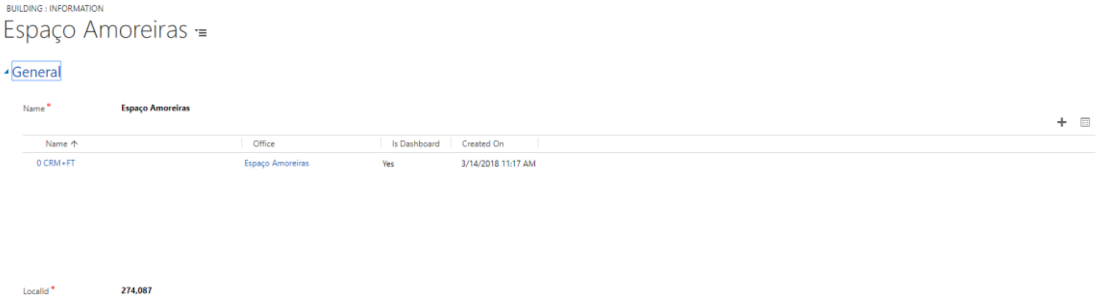
This Entity structure will work as follows:

- The Building are created.


Active Offices

Name ↑	Created On
Entrecampos CSS	9/18/2017 1:47 PM
Espaço Amoreiras	3/14/2018 11:12 AM

- Then added the Floors that are part of each building.


BUILDING : INFORMATION

Espaço Amoreiras

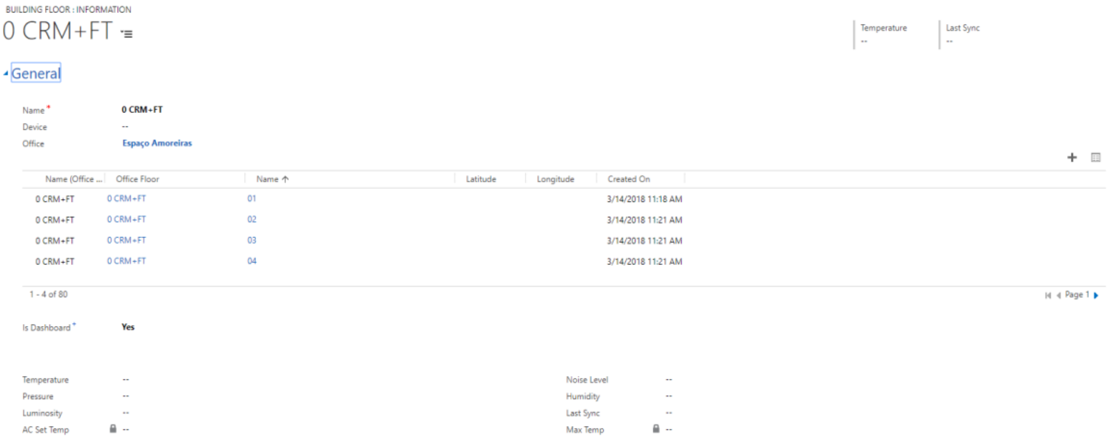
General

Name * Espaço Amoreiras

Name ↑	Office	Is Dashboard	Created On
0 CRM+FT	Espaço Amoreiras	Yes	3/14/2018 11:17 AM

LocalId * 274,087

- Finally, all the seats are created inside each Floor.



BUILDING FLOOR : INFORMATION

0 CRM+FT

Temperature	--
Last Sync	--

General

Name *	0 CRM+FT
Device	--
Office	Espaço Amoreiras

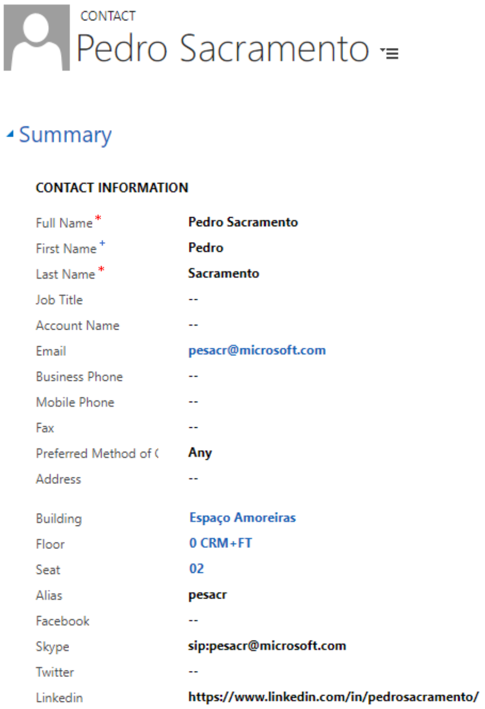
Name (Office ...	Office Floor	Name ↑	Latitude	Longitude	Created On
0 CRM+FT	0 CRM+FT	01			3/14/2018 11:18 AM
0 CRM+FT	0 CRM+FT	02			3/14/2018 11:21 AM
0 CRM+FT	0 CRM+FT	03			3/14/2018 11:21 AM
0 CRM+FT	0 CRM+FT	04			3/14/2018 11:21 AM

1 - 4 of 80

Is Dashboard *	Yes

Temperature	--	Noise Level	--
Pressure	--	Humidity	--
Luminosity	--	Last Sync	--
AC Set Temp	--	Max Temp	--

- The contacts are then assigned a Building, Floor and seat.



CONTACT

Pedro Sacramento

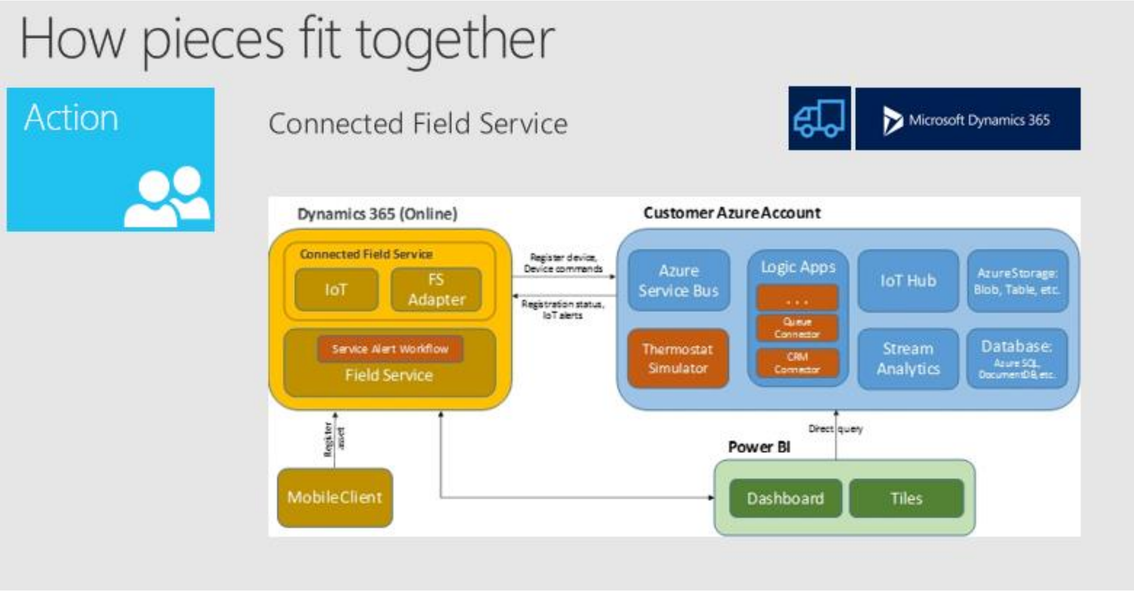
Summary

CONTACT INFORMATION

Full Name *	Pedro Sacramento
First Name +	Pedro
Last Name *	Sacramento
Job Title	--
Account Name	--
Email	pesacr@microsoft.com
Business Phone	--
Mobile Phone	--
Fax	--
Preferred Method of (	Any
Address	--
Building	Espaço Amoreiras
Floor	0 CRM+FT
Seat	02
Alias	pesacr
Facebook	--
Skype	sip:pesacr@microsoft.com
Twitter	--
Linkedin	https://www.linkedin.com/in/pedrosacramento/

We've created the logic using the contact entity, but it may use other entities instead like Users for example.

Connected Field Service



How pieces fit together

Action

Connected Field Service

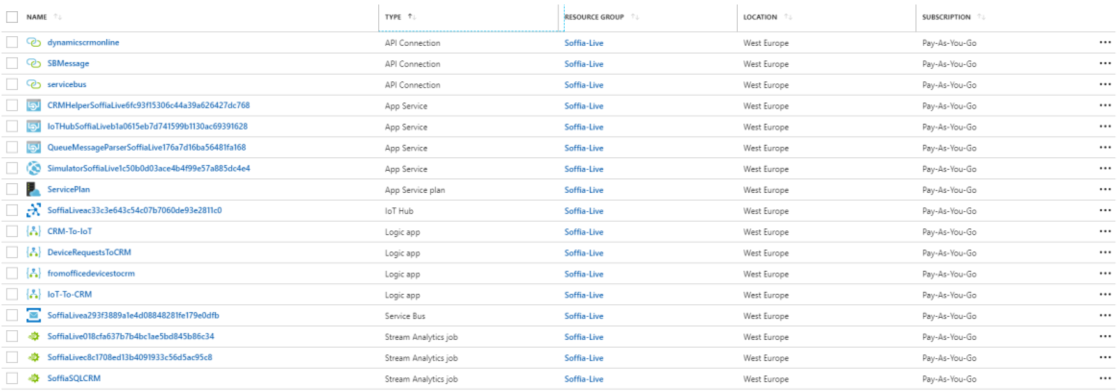
The Connected Field Service Add-on will allow an automated way of connecting Dynamics 365 for Customer Engagement with an Azure subscription, automatically deploying all Azure Stack resources that are needed for an IoT scenario.

The Add-on is available form AppSource.

By deploying the Add-on, several solutions are automatically added to the Dynamics 365 for Customer Engagement instance.

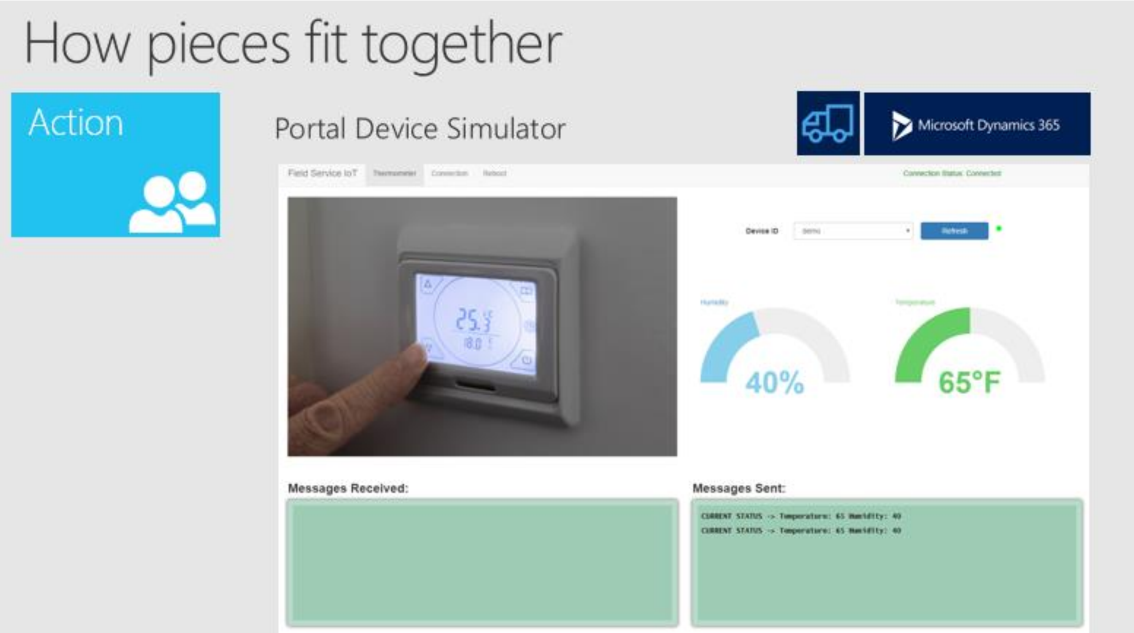
The documentation on how to deploy and configure the CFS add-on is available here. There is also an article on how to extend CFS solutions.

As mentioned before, all resources are added to the Azure Subscription.



□ NAME	TYPE	RESOURCE GROUP	LOCATION	SUBSCRIPTION	
□ dynamicscrmonline	API Connection	Soffia-Live	West Europe	Pay-As-You-Go	•••
□ SBMessage	API Connection	Soffia-Live	West Europe	Pay-As-You-Go	•••
□ servicebus	API Connection	Soffia-Live	West Europe	Pay-As-You-Go	•••
□ CRMHelperSoffiaLive6fc93f15306c44a39a626427dc768	App Service	Soffia-Live	West Europe	Pay-As-You-Go	•••
□ IoTHubSoffiaLiveb1a0615eb7d741599b1130ac69391628	App Service	Soffia-Live	West Europe	Pay-As-You-Go	•••
□ QueueMessageParserSoffiaLive176a7d16ba56481fa168	App Service	Soffia-Live	West Europe	Pay-As-You-Go	•••
□ SimulatorSoffiaLive1c50b0d03ace4b4f99e57a885dc4e4	App Service	Soffia-Live	West Europe	Pay-As-You-Go	•••
□ ServicePlan	App Service plan	Soffia-Live	West Europe	Pay-As-You-Go	•••
□ SoffiaLiveac33c3e643c54c07b7060de93e2811c0	IoT Hub	Soffia-Live	West Europe	Pay-As-You-Go	•••
□ CRM-To-IoT	Logic app	Soffia-Live	West Europe	Pay-As-You-Go	•••
□ DeviceRequestsToCRM	Logic app	Soffia-Live	West Europe	Pay-As-You-Go	•••
□ fromofficedevicestocrm	Logic app	Soffia-Live	West Europe	Pay-As-You-Go	•••
□ IoT-To-CRM	Logic app	Soffia-Live	West Europe	Pay-As-You-Go	•••
□ SoffiaLivea293f3889a1e4d08848281fe179e0dfb	Service Bus	Soffia-Live	West Europe	Pay-As-You-Go	•••
□ SoffiaLive018cfa637b7b4bc1ae5bd845b86c34	Stream Analytics job	Soffia-Live	West Europe	Pay-As-You-Go	•••
□ SoffiaLivec8c1708ed13b4091933c56d5ac95c8	Stream Analytics job	Soffia-Live	West Europe	Pay-As-You-Go	•••
□ SoffiaSQLCRM	Stream Analytics job	Soffia-Live	West Europe	Pay-As-You-Go	•••

In the resources it’s included a Web App that is a Portal Device Simulator. From the simulator it’s possible to configure the threshold and simulate the values sent from the devices.



How pieces fit together

Action

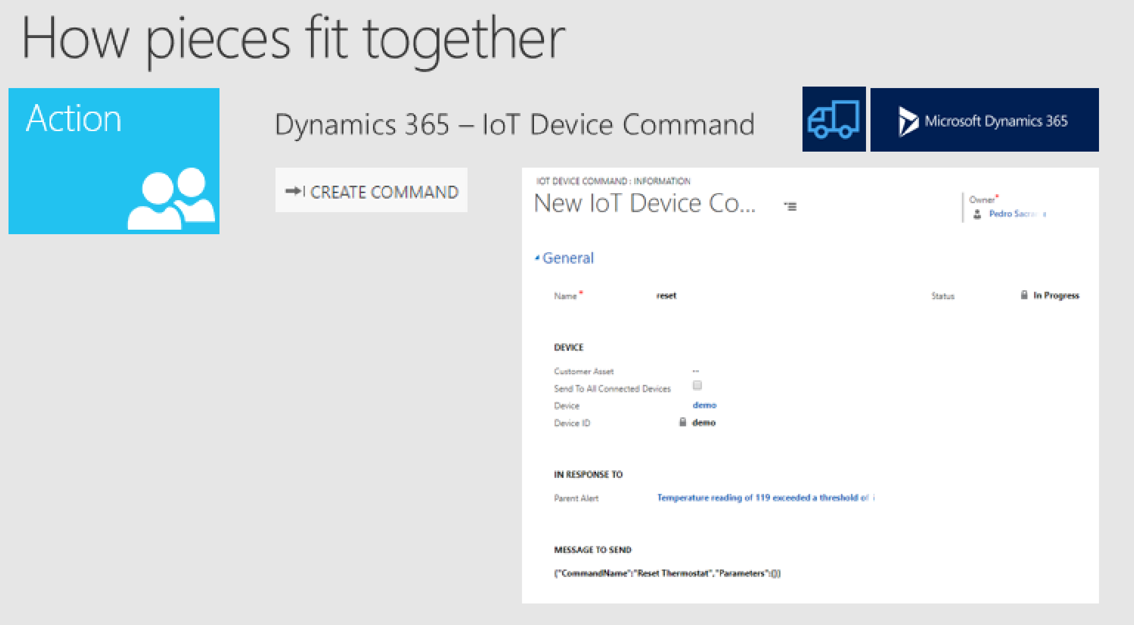
Portal Device Simulator

Messages Received:

Messages Sent:

```
CURRENT STATUS -> Temperature: 65 Humidity: 40
CURRENT STATUS -> Temperature: 65 Humidity: 40
```

The Portal Device Simulator will then create an IoT Alert in the Dynamics 365 for Customer Engagement instance if the temperature is above the threshold.



How pieces fit together

Action

Dynamics 365 – IoT Device Command

IOT DEVICE COMMAND : INFORMATION

New IoT Device Co...

Owner* Pedro Sacr

General

Name* reset

Status In Progress

DEVICE

Customer Asset --

Send To All Connected Devices □

Device demo

Device ID demo

IN RESPONSE TO

Parent Alert Temperature reading of 119 exceeded a threshold of

MESSAGE TO SEND

{"CommandName":"Reset Thermostat","Parameters":{}}

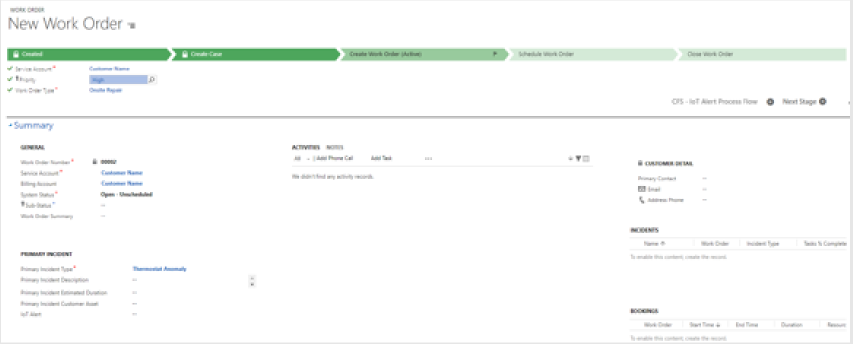
From the IoT Alert, a Work Order can be created so that the issue is investigated by the maintenance team or Field Engineers. The simulator will provide a simple and Out-of-the-box solution to test and demo an IoT scenario. The simulator will send telemetry to the Azure IoT Hub, the telemetry is then handled by the Stream Analytics that will create the IoT Alert record in the Dynamics 365 for Customer Engagement instance from the Logic App execution.

How pieces fit together

Action

Dynamics 365 – Create Work Order

Microsoft Dynamics 365



WORK ORDER

New Work Order

✓ Service Account*: Customer Name
✓ Priority: High
✓ Work Order Type*: Onsite Repair

CFS - IoT Alert Process Flow ⚙ Next Stage

Summary

GENERAL

Work Order Number*: 00002
Service Account*: Customer Name
Billing Account: Customer Name
System Status*: Open - Unscheduled
Sub-Status*: --
Work Order Summary: --

PRIMARY INCIDENT

Primary Incident Type*: Thermostat Anomaly
Primary Incident Description: --
Primary Incident Estimated Duration: --
Primary Incident Customer Asset: --
IoT Alert: --

ACTIVITIES NOTES

We didn't find any activity records.

CUSTOMER DETAIL

Primary Contact: --
Email: --
Address Phone: --

INCIDENTS

Name	Work Order	Incident Type	Tasks % Complete

To enable this content, create the record.

BOOKINGS

Work Order	Start Time	End Time	Duration	Resourc

To enable this content, create the record.

Microsoft Azure

IoT Hub

The Azure IoT Hub is the main resource in the architecture as it’s the piece responsible for receiving all telemetry from the Devices.

In some scenarios, were the level of telemetry sent from the devices is very low, we could bypass the Azure Stack and connect devices directly to our Line of Business Application.

The Azure IoT Hub will help handling big volumes of data in multiple threads from multiple devices and assure that there isn’t any telemetry lost. This is the best scenario for scalable solutions with big data volumes.

For more information see the following article:

https://azure.microsoft.com/en-us/services/iot-hub/

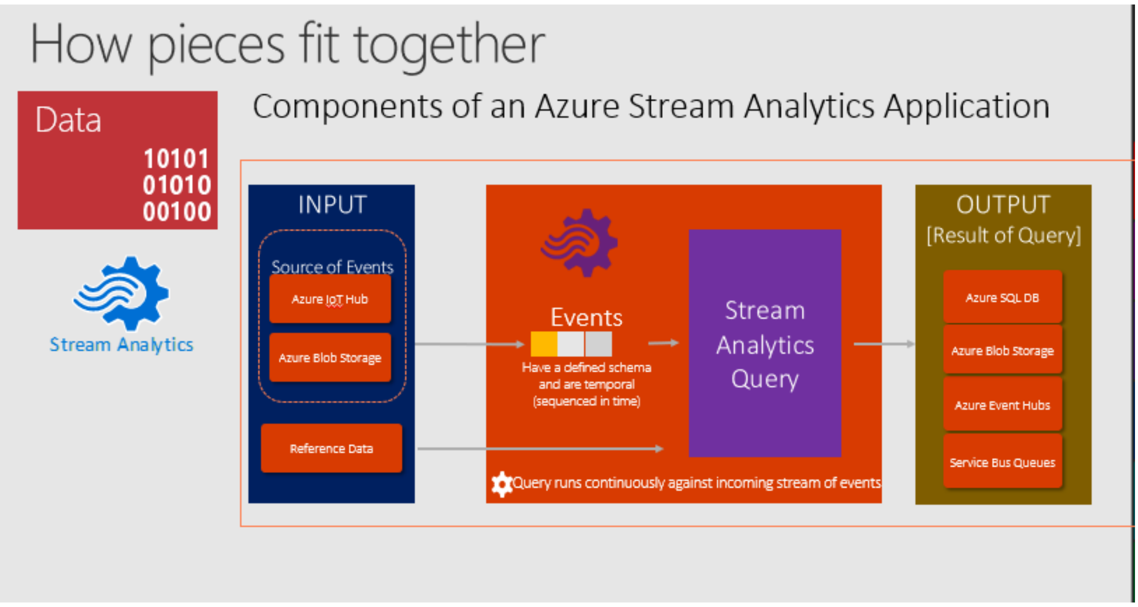
Stream Analytics

The Stream Analytics integrated with Azure IoT Hub and the remaining Azure Stack to enable powerful real-time analytics on data that is collected from the devices.

This is our center piece in the architecture as it will turn data into insights.

For more information see the following article:

https://azure.microsoft.com/en-us/services/stream-analytics/



How pieces fit together

Components of an Azure Stream Analytics Application

Data

10101
01010
00100

Stream Analytics

INPUT

Source of Events

Azure IoT Hub

Azure Blob Storage

Reference Data

Events

Have a defined schema and are temporal (sequenced in time)

Stream Analytics Query

Query runs continuously against incoming stream of events

OUTPUT

[Result of Query]

Azure SQL DB

Azure Blob Storage

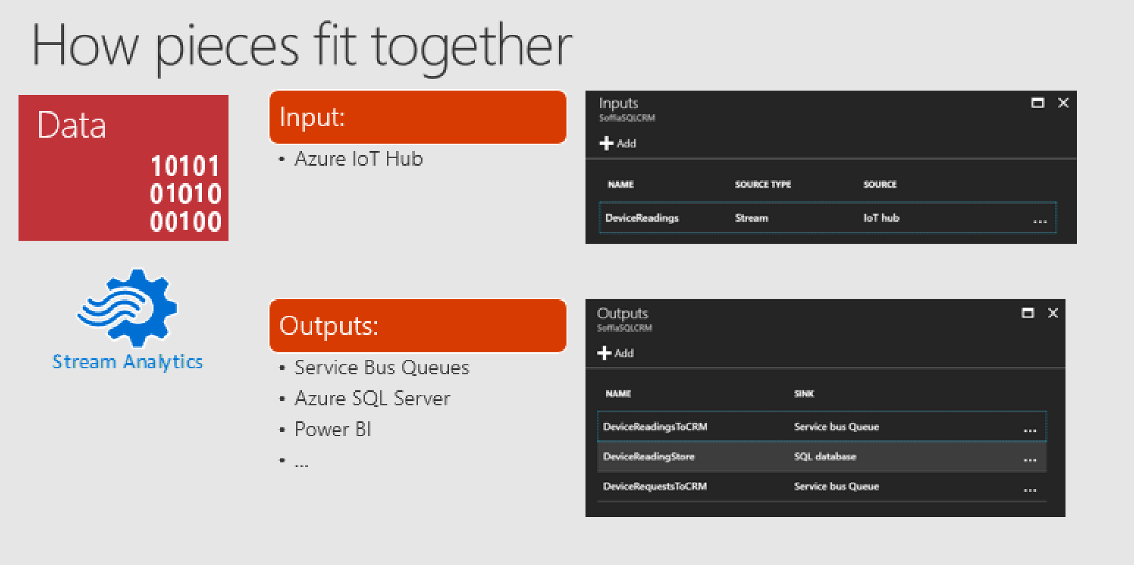
Azure Event Hubs

Service Bus Queues

In the Stream Analytics we’ll define what happens with the data collected from the devices and sent to the Azure IoT Hub.

In our scenario, data is stored in a SQL Azure Database for later analysis and reporting.

With the usage of Service Bus Queues and Logic Apps, the information is also updated in Dynamics 365 for Customer Engagement, in the “Building Floor” entity, so that the information is visible from the custom portal to all users.



How pieces fit together

Data

10101
01010
00100

Stream Analytics

Input:

- Azure IoT Hub

Inputs
SoffiaSQLCRM
+ Add

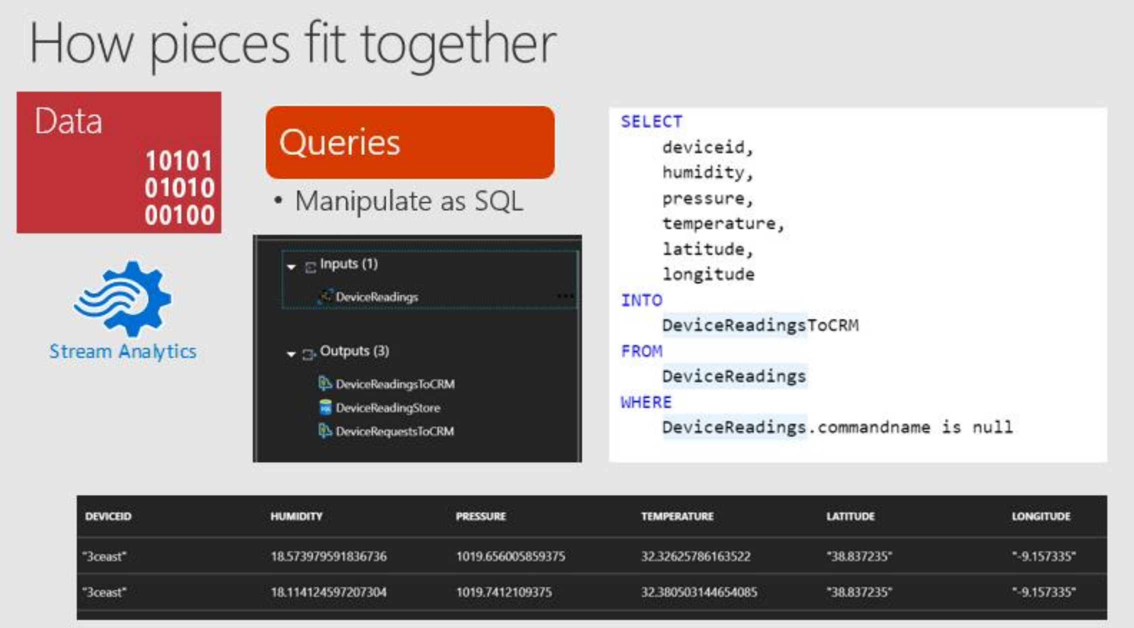
NAME	SOURCE TYPE	SOURCE
DeviceReadings	Stream	IoT hub

Outputs:

- Service Bus Queues
- Azure SQL Server
- Power BI
- ...

Outputs
SoffiaSQLCRM
+ Add

NAME	SINK
DeviceReadingsToCRM	Service bus Queue
DeviceReadingStore	SQL database
DeviceRequestsToCRM	Service bus Queue



How pieces fit together

Data

10101
01010
00100

Stream Analytics

Queries

- Manipulate as SQL

```
Inputs (1)
    DeviceReadings
Outputs (3)
    DeviceReadingsToCRM
    DeviceReadingStore
    DeviceRequestsToCRM
```

```
SELECT
    deviceid,
    humidity,
    pressure,
    temperature,
    latitude,
    longitude
INTO
    DeviceReadingsToCRM
FROM
    DeviceReadings
WHERE
    DeviceReadings.commandname is null
```

DEVICEID	HUMIDITY	PRESSURE	TEMPERATURE	LATITUDE	LONGITUDE
"3ceast"	18.573979591836736	1019.656005859375	32.32625786163522	"38.837235"	"-9.157335"
"3ceast"	18.114124597207304	1019.7412109375	32.380503144654085	"38.837235"	"-9.157335"

Documentation and articles regarding Stream Analytics:

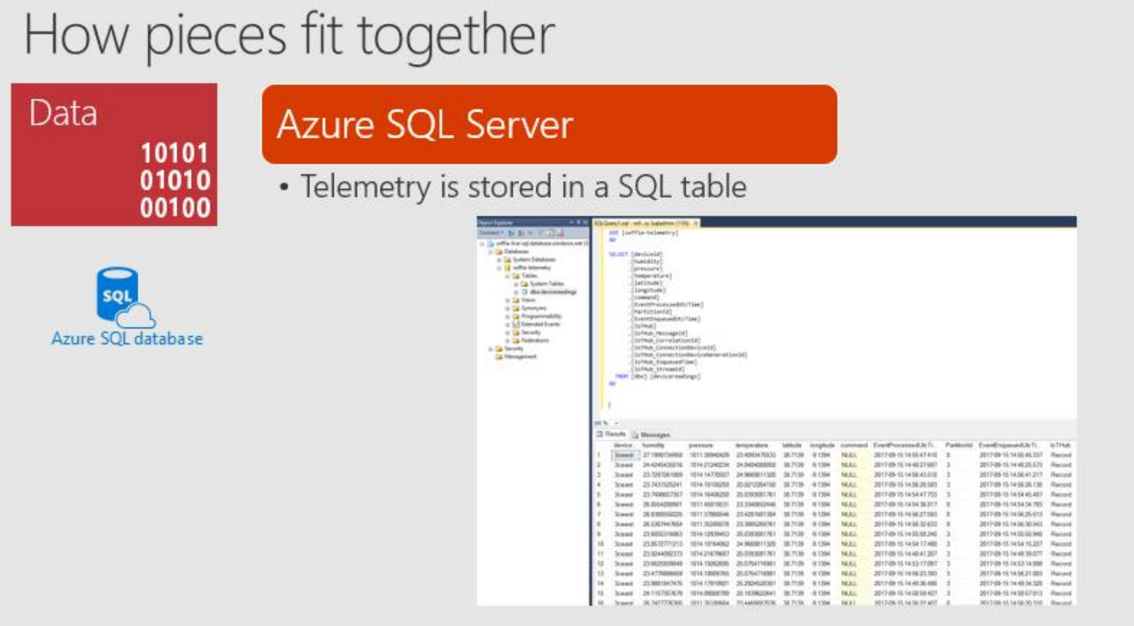
https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-build-an-iot-solution-using-stream-analytics

https://azure.microsoft.com/en-us/resources/videos/handling-iot-data-with-azure-stream-analytics/

https://community.dynamics.com/crm/b/mannygrewaldynamicscrm/archive/2017/07/01/azure-iot-hub-streaming-analytics-simulator https://blogs.msdn.microsoft.com/david/2016/09/12/learning-guide-for-azure-stream-analytics/

https://danielburtcrm.wordpress.com/2016/12/05/create-connected-service-cases-in-dynamics-365-using-an-iot-device-and-azure/

Azure SQL Database



How pieces fit together

Azure SQL Server

- Telemetry is stored in a SQL table

```
USE [soffia-telemetry]
GO

SELECT [deviceId]
      ,[humidity]
      ,[pressure]
      ,[temperature]
      ,[latitude]
      ,[longitude]
      ,[command]
      ,[EventProcessedUtcTime]
      ,[PartitionId]
      ,[EventEnqueuedUtcTime]
      ,[IoTHub]
      ,[IoTHub_MessageId]
      ,[IoTHub_CorrelationId]
      ,[IoTHub_ConnectionDeviceId]
      ,[IoTHub_ConnectionDeviceGenerationId]
      ,[IoTHub_EnqueuedTime]
      ,[IoTHub_StreamId]
  FROM [dbo].[devicereadings]
GO
```

	device	humidity	pressure	temperature	latitude	longitude	command	EventProcessedUtcTi...	PartitionId	EventEnqueuedUtcTi...	IoTHub
1	[illegible]	[illegible]	[illegible]	[illegible]	38.7139	-9.1394	NULL	[illegible]	0	[illegible]	Record
2	[illegible]	[illegible]	[illegible]	[illegible]	38.7139	-9.1394	NULL	[illegible]	3	[illegible]	Record
3	[illegible]	[illegible]	[illegible]	[illegible]	38.7139	-9.1394	NULL	[illegible]	3	[illegible]	Record
4	[illegible]	[illegible]	[illegible]	[illegible]	38.7139	-9.1394	NULL	[illegible]	3	[illegible]	Record
5	[illegible]	[illegible]	[illegible]	[illegible]	38.7139	-9.1394	NULL	[illegible]	3	[illegible]	Record
6	[illegible]	[illegible]	[illegible]	[illegible]	38.7139	-9.1394	NULL	[illegible]	0	[illegible]	Record
7	[illegible]	[illegible]	[illegible]	[illegible]	38.7139	-9.1394	NULL	[illegible]	0	[illegible]	Record
8	[illegible]	[illegible]	[illegible]	[illegible]	38.7139	-9.1394	NULL	[illegible]	0	[illegible]	Record
9	[illegible]	[illegible]	[illegible]	[illegible]	38.7139	-9.1394	NULL	[illegible]	3	[illegible]	Record
10	[illegible]	[illegible]	[illegible]	[illegible]	38.7139	-9.1394	NULL	[illegible]	3	[illegible]	Record
11	[illegible]	[illegible]	[illegible]	[illegible]	38.7139	-9.1394	NULL	[illegible]	3	[illegible]	Record
12	[illegible]	[illegible]	[illegible]	[illegible]	38.7139	-9.1394	NULL	[illegible]	3	[illegible]	Record
13	[illegible]	[illegible]	[illegible]	[illegible]	38.7139	-9.1394	NULL	[illegible]	3	[illegible]	Record
14	[illegible]	[illegible]	[illegible]	[illegible]	38.7139	-9.1394	NULL	[illegible]	3	[illegible]	Record
15	[illegible]	[illegible]	[illegible]	[illegible]	38.7139	-9.1394	NULL	[illegible]	3	[illegible]	Record
16	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NULL	[illegible]	0	[illegible]	Record

In the SQL Azure database, the telemetry collected by the devices is stored. The data is manipulated as a query by the Stream Analytics.

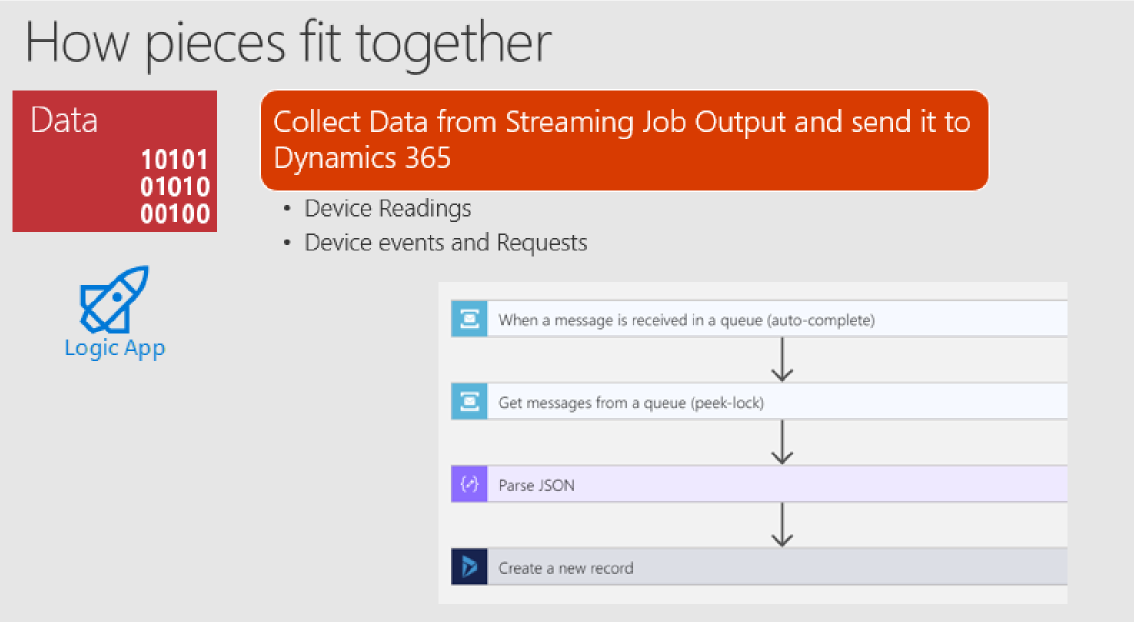
Storing this information will not only allow us to build reporting but also to future analyze the information with Machine Learning.

Logic Apps

The Logic Apps piece in the Azure Stack will be responsible for creating the information in Dynamics or sending information from Dynamics back to the device.

The Connected Field Service Add-on bring out-of-the-box Logic Apps that can be adjusted to the requirements.

In our scenario we have the Logic App that will update the information in the "Building Floor" entity.



How pieces fit together

Data

Collect Data from Streaming Job Output and send it to Dynamics 365

- Device Readings
- Device events and Requests

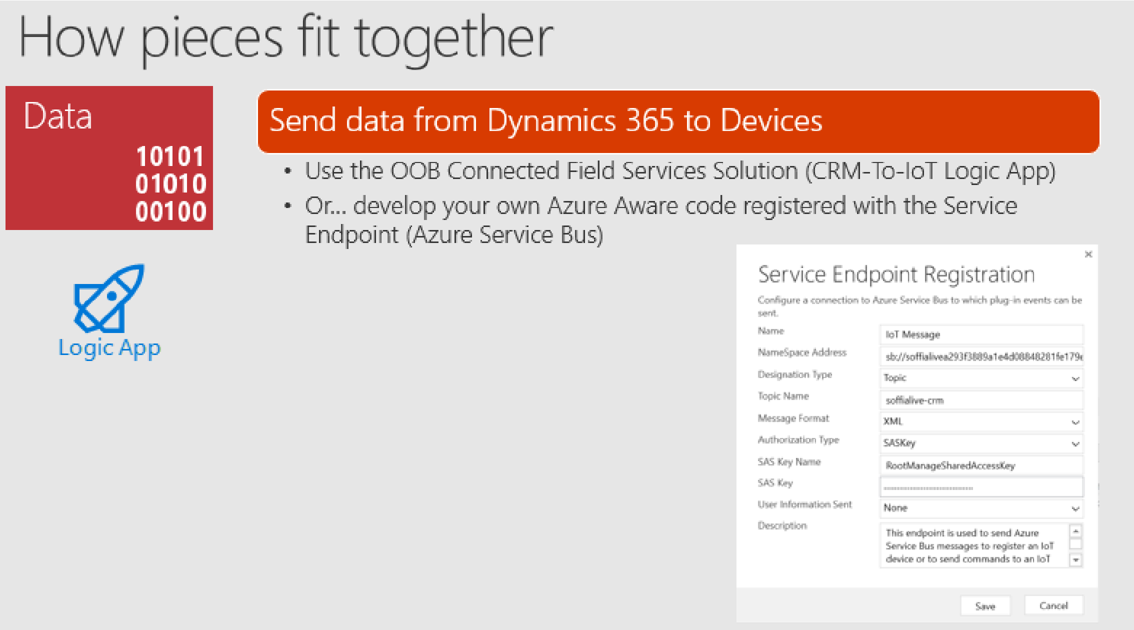
When a message is received in a queue (auto-complete)

Get messages from a queue (peek-lock)

Parse JSON

Create a new record

And the Logic App that will trigger an action in the device that is generated from the Field Service solution. In this case the device is requested to reboot, using this mechanism it could be possible to deploy updates on the device configuration (update thresholds for example) or pushing code updates.



How pieces fit together

Data

Send data from Dynamics 365 to Devices

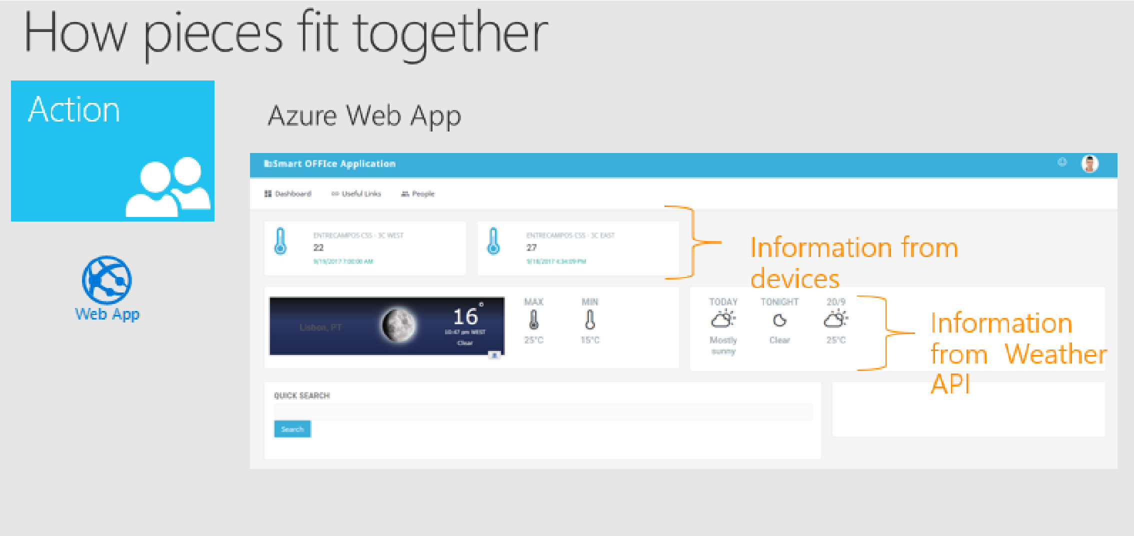
- Use the OOB Connected Field Services Solution (CRM-To-IoT Logic App)
- Or... develop your own Azure Aware code registered with the Service Endpoint (Azure Service Bus)

Service Endpoint Registration

Configure a connection to Azure Service Bus to which plug-in events can be sent.

Name	IoT Message
NameSpace Address	sb://soffialivea293f3889a1e4d08848281fe179e
Designation Type	Topic
Topic Name	soffialive-crm
Message Format	XML
Authorization Type	SASKey
SAS Key Name	RootManageSharedAccessKey
SAS Key	•••••••••••••••••••••••••••••
User Information Sent	None
Description	This endpoint is used to send Azure Service Bus messages to register an IoT device or to send commands to an IoT

Azure WebApp



How pieces fit together

Action

Web App

Azure Web App

Information from devices

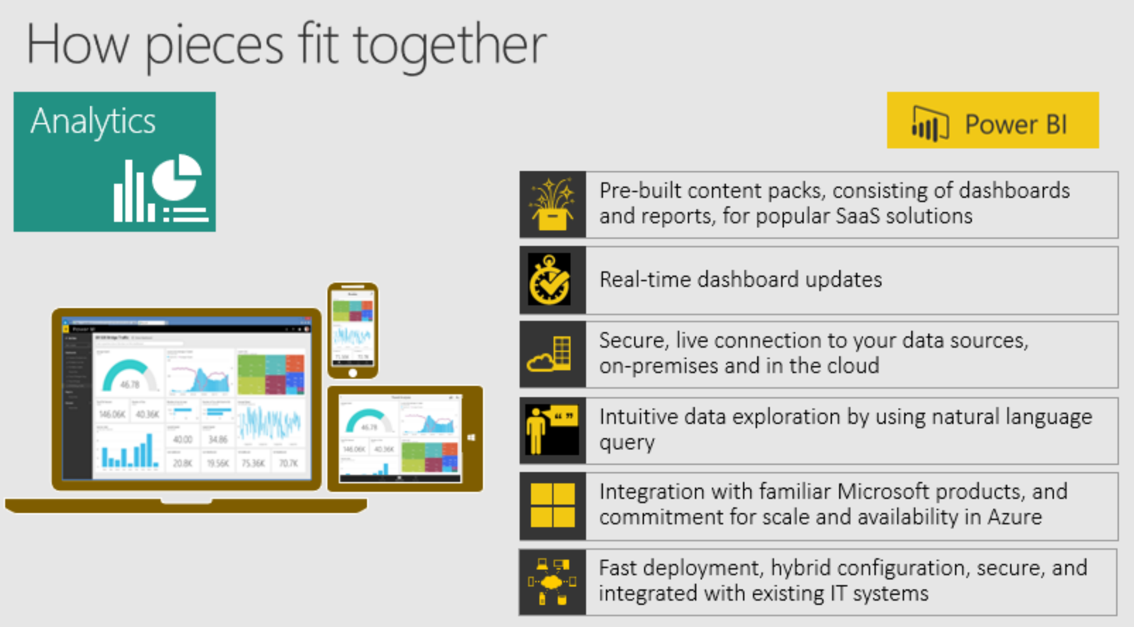
Information from Weather API

The temperature collected from the devices is displayed to the users from a custom Web Portal created using Azure Web Apps. The Portal will connect to Dynamics and display the information and allow the users (contacts in Dynamics 365 for Customer Engagement) to edit their profiles and configure their seating place in the Building.

This will also allow for the users to know where a specific person is located in the Building/floor.

From the Portal is would also be possible for users to provide feedback regarding the temperature or other information that could be important related with the wellbeing at work. Capturing this information would allow additional measure to be taken and also analysis in terms of what is being captured by the devices and the feedback being provided by users.

PowerBI



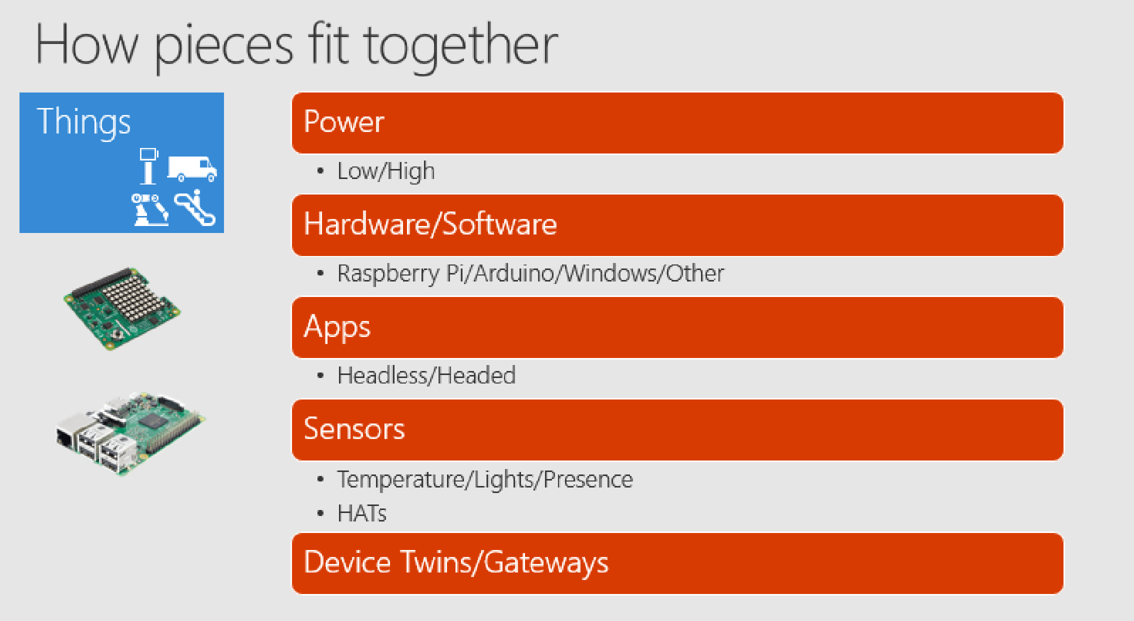
How pieces fit together

Analytics
Power BI

- Pre-built content packs, consisting of dashboards and reports, for popular SaaS solutions
- Real-time dashboard updates
- Secure, live connection to your data sources, on-premises and in the cloud
- Intuitive data exploration by using natural language query
- Integration with familiar Microsoft products, and commitment for scale and availability in Azure
- Fast deployment, hybrid configuration, secure, and integrated with existing IT systems

Since all information collected from the devices is stored in a SQL Azure database, by using PowerBI we can create dashboards and reports and make them available either from PowerBI directly or from the Dynamics 365 for Customer Engagement instance.

Devices



How pieces fit together

Things

Power
- Low/High

Hardware/Software
- Raspberry Pi/Arduino/Windows/Other

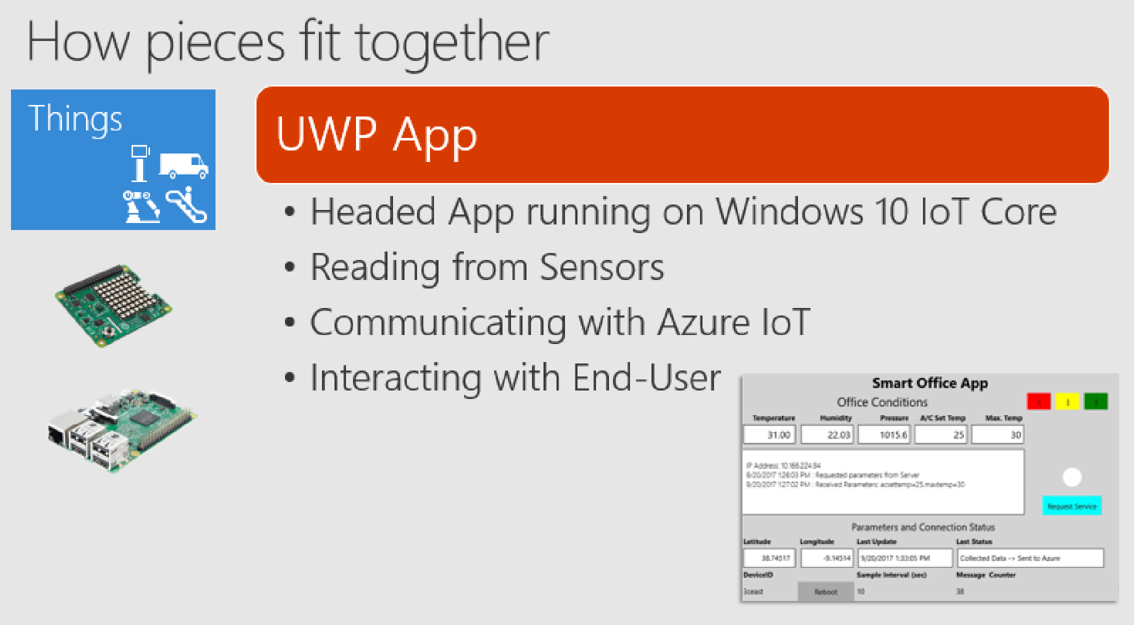
Apps
- Headless/Headed

Sensors
- Temperature/Lights/Presence
- HATs

Device Twins/Gateways

The device used for this scenario was a Raspberry PI. The below articles will explain how to build a solution to measure temperature or other information.

In this case the device has an embedded screen which allows to run an Universal Windows Application that will display the information that is being collected by the sensors in real-time.



How pieces fit together

Things

UWP App
- Headed App running on Windows 10 IoT Core
- Reading from Sensors
- Communicating with Azure IoT
- Interacting with End-User

Smart Office App

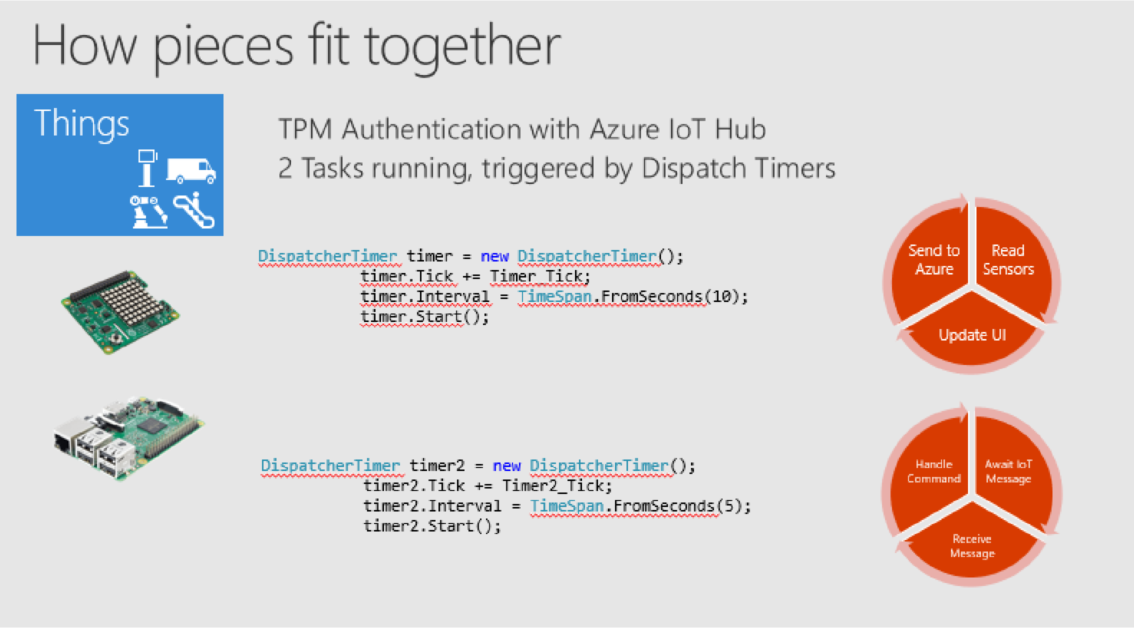
Office Conditions

Temperature: 31.00
Humidity: 22.03
Pressure: 1015.6
A/C Set Temp: 25
Max. Temp: 30

IP Address: 10.166.224.84
9/20/2017 12:6:03 PM : Requested parameters from Server
9/20/2017 12:7:02 PM : Received Parameters: acsettemp=25,maxtemp=30

Parameters and Connection Status

Latitude: 38.74517
Longitude: -9.14514
Last Update: 9/20/2017 1:33:05 PM
Last Status: Collected Data -> Sent to Azure
DeviceID: 3ceast
Sample Interval (sec): 10
Message Counter: 38



How pieces fit together

Things

TPM Authentication with Azure IoT Hub
2 Tasks running, triggered by Dispatch Timers

```
DispatcherTimer timer = new DispatcherTimer();
            timer.Tick += Timer_Tick;
            timer.Interval = TimeSpan.FromSeconds(10);
            timer.Start();
```

Send to Azure
Read Sensors
Update UI

```
DispatcherTimer timer2 = new DispatcherTimer();
            timer2.Tick += Timer2_Tick;
            timer2.Interval = TimeSpan.FromSeconds(5);
            timer2.Start();
```

Handle Command
Await IoT Message
Receive Message

For more information on how to build the code to run from the devices see the following articles:

https://docs.microsoft.com/pt-pt/azure/iot-hub/iot-hub-csharp-csharp-getstarted

https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-device-management-overview

http://gunnarpeipman.com/2016/01/measuring-temperature-with-windows-10-iot-core-and-raspberry-pi/

https://jeremylindsayni.wordpress.com/2016/06/09/how-to-measure-temperature-using-c-raspberry-pi-windows-10-iot-core-and-the-adafruit-mcp9808-sensor/

https://www.hackster.io/peejster/monitor-your-room-s-temp-and-humidity-from-anywhere-4a0106

Application Management

https://github.com/ms-iot/iot-core-azure-dm-client/blob/develop/docs/application-management.md

Windows Update Management

https://github.com/ms-iot/iot-core-azure-dm-client/blob/develop/docs/windows-update-management.md