Microsoft

Thermoplan uses Azure to manage data for its smart coffee machine

Ronnie Saurenmann, Ken Casada - Jun 7, 2017

Thermoplan develops and produces professional coffee machines. The new Internet-connected models need to be remotely managed, constantly tracked for their usage, and able to implement predictive maintenance techniques. The solution consists of an Azure IoT Hub acting as a telemetry data collector of product statistics as well as an asset manager and software and configuration update system. Azure Stream Analytics is used to analyze and distribute the data in real time, including statistics about the amount and types of coffee produced, and cleaning cycles vs. recommended intervals. The solution relies on Azure Data Lake to efficiently store, filter, and analyze the huge quantity of data being produced by tens of thousands of coffee machines, up to 200 GB of telemetry data per day.

Key technologies used

- Microsoft Azure Data Factory
- Azure Data Lake
- Azure Event Hubs
- Azure IoT Hub
- Azure Notification Hubs
- Azure SQL Database
- Azure Storage
- Azure Stream Analytics
- Microsoft Power BI
- Web Apps feature of Azure App Service

Core team

Thermoplan AG:

- Rolf Hochstrasser – Project Manager and Product Owner
- Philipp Roebrock – Head Software Engineering
- Andreas Heimberger – Senior Software Engineer (Device)

bbv Software Services:

- Sergio Filosofo – Senior Project Leader
- Roland Krummenacher – Senior Software Architect
- Marion Frei – Senior Software Engineer (Cloud)
- Samuel Schwager – Senior Software Engineer (Cloud)
- Dave Buechi – Senior Software Engineer (Device)

Microsoft Switzerland:

- Ronnie Saurenmann – Principal Technical Evangelist
- Ken Casada – Senior Technical Evangelist

Customer profile

Thermoplan of Weggis, Switzerland, develops and produces professional coffee machines that enjoy a great reputation all over the world for their reliability, functionality, and their unique features and benefits. By signing long-term supply contracts with renowned international companies such as Starbucks, Statoil, Nespresso, and Costa Coffee, Thermoplan has become one of the market leaders in the business segment of fully automatic coffee machines for the food service and catering industry.

bbv Software Services of Lucerne, Switzerland, is a software and consulting company. They are focused on contributing their expertise to some of the most outstanding visions, projects, and challenges of their clients. They are tasked with creating the cloud solution for Thermoplan.

Problem statement

Thermoplan is bringing to market a new model of their most popular coffee machine, which has tens of thousands of units deployed worldwide. Nowadays for the topology of customers that buy this type of coffee machine, the need to remotely manage, constantly track usage, and do predictive maintenance is of paramount importance; that's the reason why the new machines are connected to the Internet.

Thermoplan is building a solution that takes care of three central aspects:

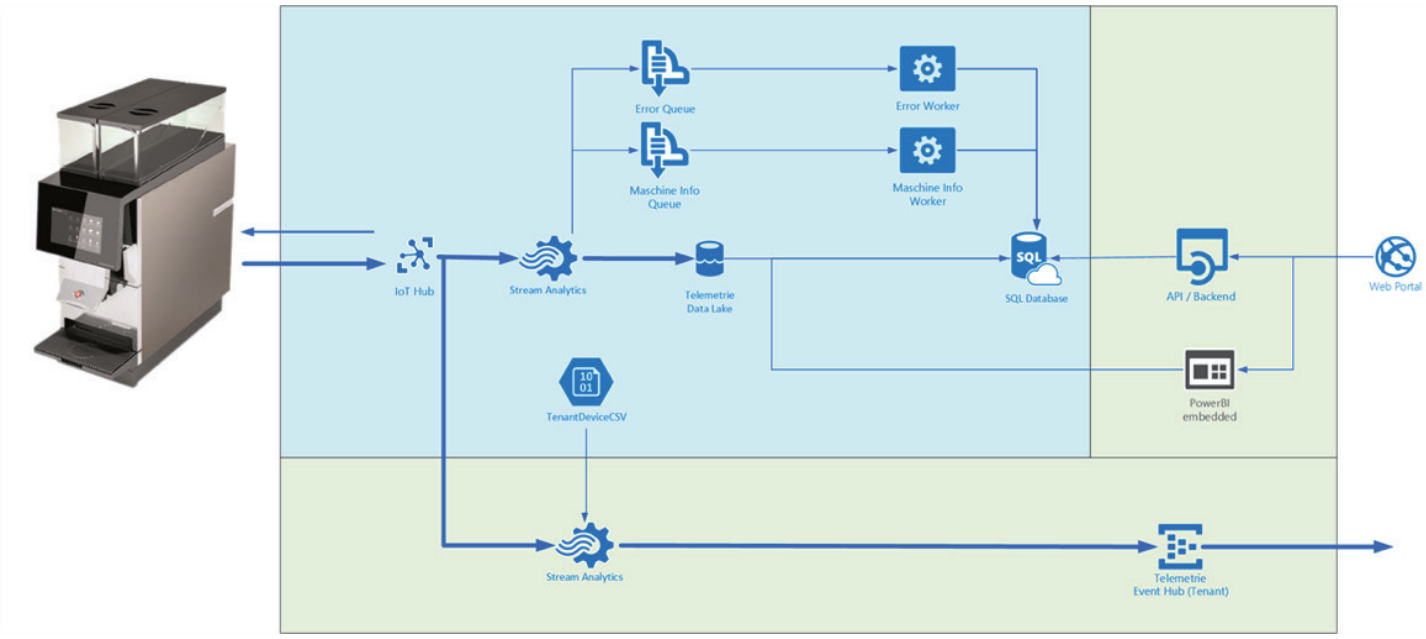
- Tracking the production of coffee in terms of quantity and quality, which allows the customer to monitor the production of coffee across all locations, gaining critical insights about the process.
- Maintenance of coffee machines. Thanks to telemetry data, Thermoplan and its customers have the ability to monitor the reliability of their coffee machines and at the same time optimize the predictive maintenance process.
- Deploying software updates efficiently and cost effectively. This remote update functionality will enable new product recipes (such as a different set of parameters for coffee and milk).

Therefore, Thermoplan needs a reliable, scalable, and cost effective solution to ingest, store, and process this large quantity of telemetry data coming in real time from this new coffee machine, which includes forwarding the data to its customers for further analysis.

Solution and steps

The proposed solution consists of a secure bi-directional communication between the coffee machine and the cloud through Azure IoT Hub. The telemetry data is then analyzed through Azure Stream Analytics and distributed according to its content to various destinations such as alarm queues, Azure Data Lake, or Azure Event Hubs for end-customer data processing. Moreover, through APIs and Power BI Embedded, a web portal is being constructed to visualize status and key production metrics.

Architecture diagram



Technical delivery

Security details

Security is a key concern for Thermoplan. A top priority is to exclude any potential data leaks and to ensure that remote commands such as software updates are not hackable.

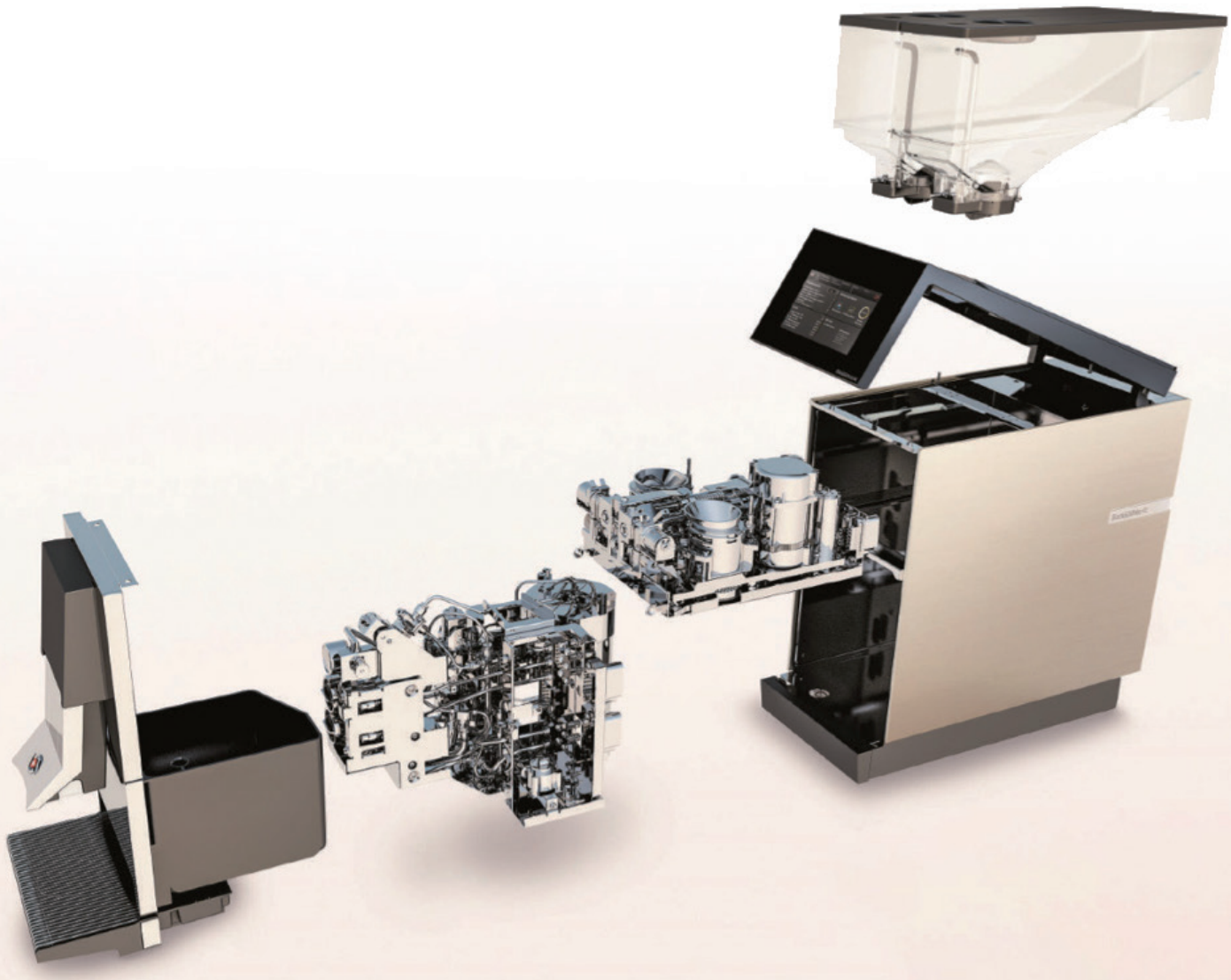
By making use of the security capabilities of IoT Hub, we can securely authenticate every single coffee machine and securely transmit data in both directions. The idea is to use the Advanced Message Queuing Protocol (AMQP) to optimize the communication channel and provide efficient bi-directional communication without incurring an HTTP polling limitation. AMQP over WebSockets communication is encrypted through Transport Layer Security (TLS).

For the authentication part, we rely on the authentication and authorization mechanism offered by IoT Hub, which is implemented in the following way:

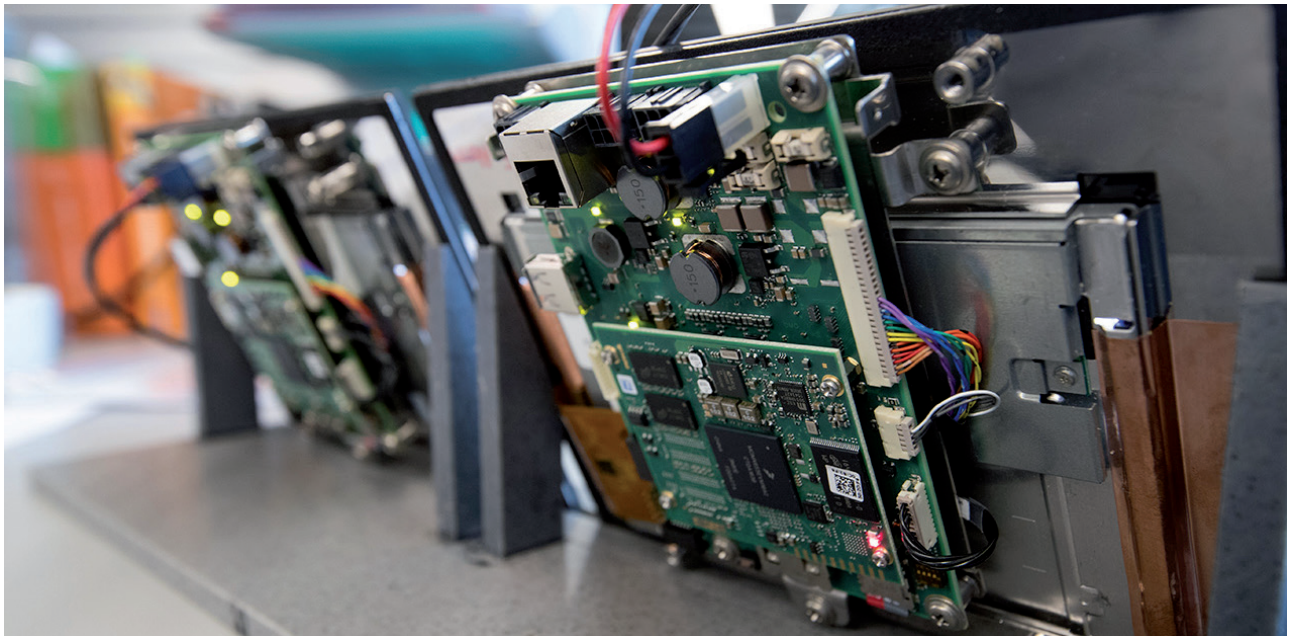
- A unique primary key is issued to and locally installed on every coffee machine. This key is securely stored within the coffee machine and used to generate SAS tokens that are then used in every communication between the coffee machine and IoT Hub.
- This ensures that the machine is the only recipient of the addressed message and that nobody can send malicious messages.
- This solution allows IoT Hub to uniquely identify the specific machine sending the telemetry message.
- In the future, we would like to further increase security by using the X.509 client certificate authentication mechanism that is supported out-of-the-box by IoT Hub.

Device used

The new generation of the coffee machine has an embedded computer with a custom Linux-based system (Linux kernel 4.1.36) created by using the Yocto Project. The embedded system is powered by a Freescale i.MX6 single core CPU. A standard machine model contains 2 brushless DC motors, 3 step motors, 5 temperature sensors, 1 pressure sensor, and multiple valves and heaters.



The embedded system collects data from all of these sensors by using the system bus. The system code is written in C++. The coffee machines are connected to the network through an Ethernet port. In the future, it will also be possible to plug in a MiFi-style USB dongle in case a wired connection is not possible.



Device messages sent

We use JSON as a message format, and we have different types of messages. The following sections provide the description, scope, and frequency of each message.

MachineInfo

The first type of message, `MachineInfo`, is a ping-home message that is sent once a day by every coffee machine. It contains basic information such as software version, hardware module IDs, language, location, and network interface.

This message supports the assets management process. As you can see from the following sample, this message is quite small.


```
    {
        "type": "MachineInfo",
        "version": "1.5.0",
        "eventtimelocal": "2017-03-14T18:21",
        "eventtime": "2017-03-14T16:21Z",
        "body": {
                "location": {
                        "timezone": "Switzerland/Bern",
                        "utcoffset": "+02:00",
                        "countrycode": "CH",
                        "language": "en"
                },
                "software": {
                        "version": "1.0.0"
                },
                "modules": [{
                        "name": "steamModule",
                        "id": "0x12345"
                }, {
                        "name": "mechanicModuleLeft",
                        "id": "0x34567"
                }],
                "networkInterfaces": [{
                        "name": "eth0",
                        "address": "13-82-46-A4-33-57"
                }, {
                        "name": "wlan0",
                        "address": "AB-B3-13-E9-9C-E1"
```

```
                }]
        }
    }
```

ProductResult (not a complete sample)

The second type of message, `ProductResult`, is the most important. This message is generated and sent to IoT Hub every time the machine produces a product. It contains very detailed information about the coffee being made, including the recipe, the amount of coffee and water being used, the duration of the entire process, and the steam temperature. Thanks to all this information, we are able to not only track the type and amount of product being used, but also determine the quality of the product compared to its recipe. To give you an idea, the size of this message is around 2 KB; consider that a single machine can produce between 500 and 1000 cups of coffee per day.

```
    {
        "type": "ProductResult",
        "version": "1.5.0",
        "eventtimelocal": "2017-03-14T18:21",
        "eventtime": "2017-03-14T16:21Z",
        "body": {
                "name": "",
                "id": "Double_Espresso_Regular_NotUpdosed_Undefined_
                "Recipe": {
                        "Category": "Espresso",
                        "IngredientParametersCoffee": {
                                "Valid": true,
                                "Name": "",
                                "Module": "CoffeeModule1",
                                "Success": true,
                                "Grinder1": {
                                        "Valid": true,
                                        "GrinderIndex": 1,
                                        "CoffeeAmountGrams": 14
                                        ..........................
```

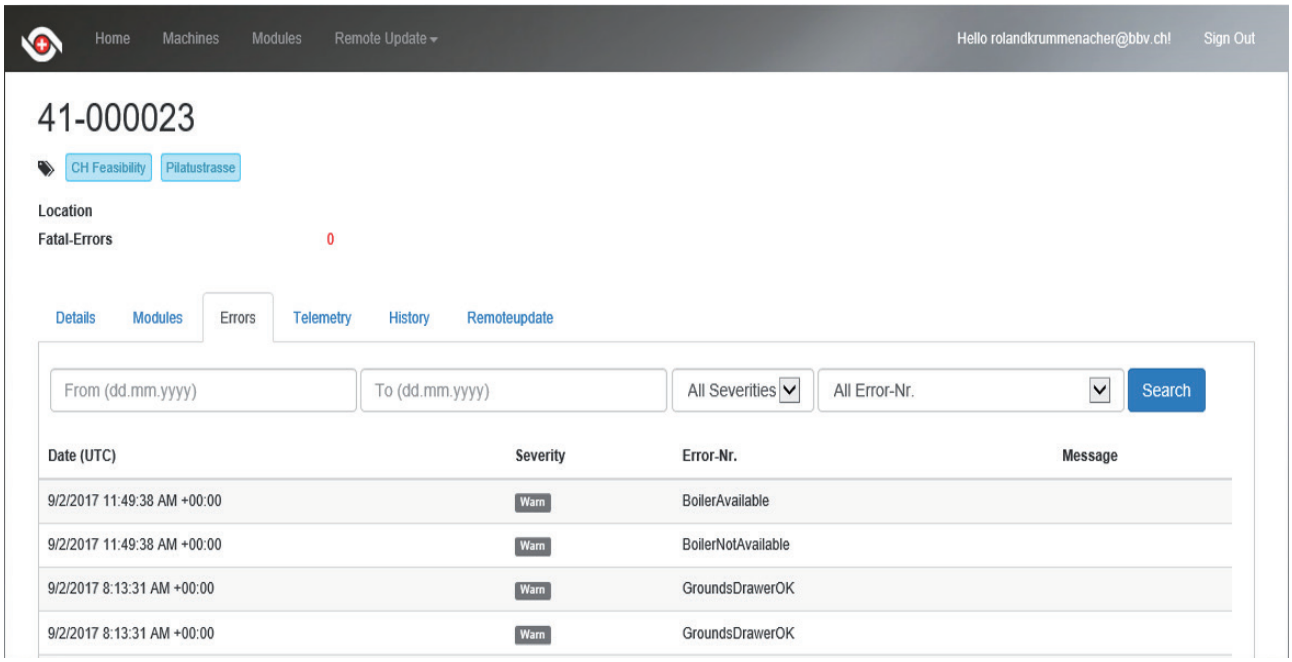
```
                },
                "Result": {
                        "Success": true,
                        "IngredientResultCoffee": {
                                "Valid": true,
                                "Name": "",
                                "Module": "CoffeeModule1",
                                "Success": true,
                                "ErrorNumber": 0,
                                "BrewChamberIndex": 0,
                                "Grinder1": {
                                        "Valid": true,
                                        "GrinderIndex": 1,
                                        "CoffeeAmountGrams": 14,
                                        "GrinderRateGramsPerSecond":
                                        ..........................
                }
        }
    }
```

Error

The third type of message, `Error`, is related to potential errors being generated by a coffee machine. This message is only sent in case of errors and is redirected to a dedicated error queue that triggers notifications.

```
    {
        "type": "Error",
        "version": "1.5.0",
        "eventtimelocal": "2017-03-14T18:21",
        "eventtime": "2017-03-14T16:21Z",
        "body": {
                "severity": 3,
                "number": 31,
                "stateful": false,
                "index": 0,
                "module": "RunlevelManager",
                "source": "Devices",
                „internaltext“: „Mainboard: First ping failed (ReceiveError)“
}       }
    }
```

Report showing error messages



Finally, there are message types for triggering and monitoring the firmware update process.

RemoteUpdate

This is a cloud-to-device message type.

```
    {
        "type": "RemoteUpdate",
        "version": "1.5.0",
        "body": {
                "jobId": 1547,
                "storageUri": "RemoteUpdate"
        }
    }
```

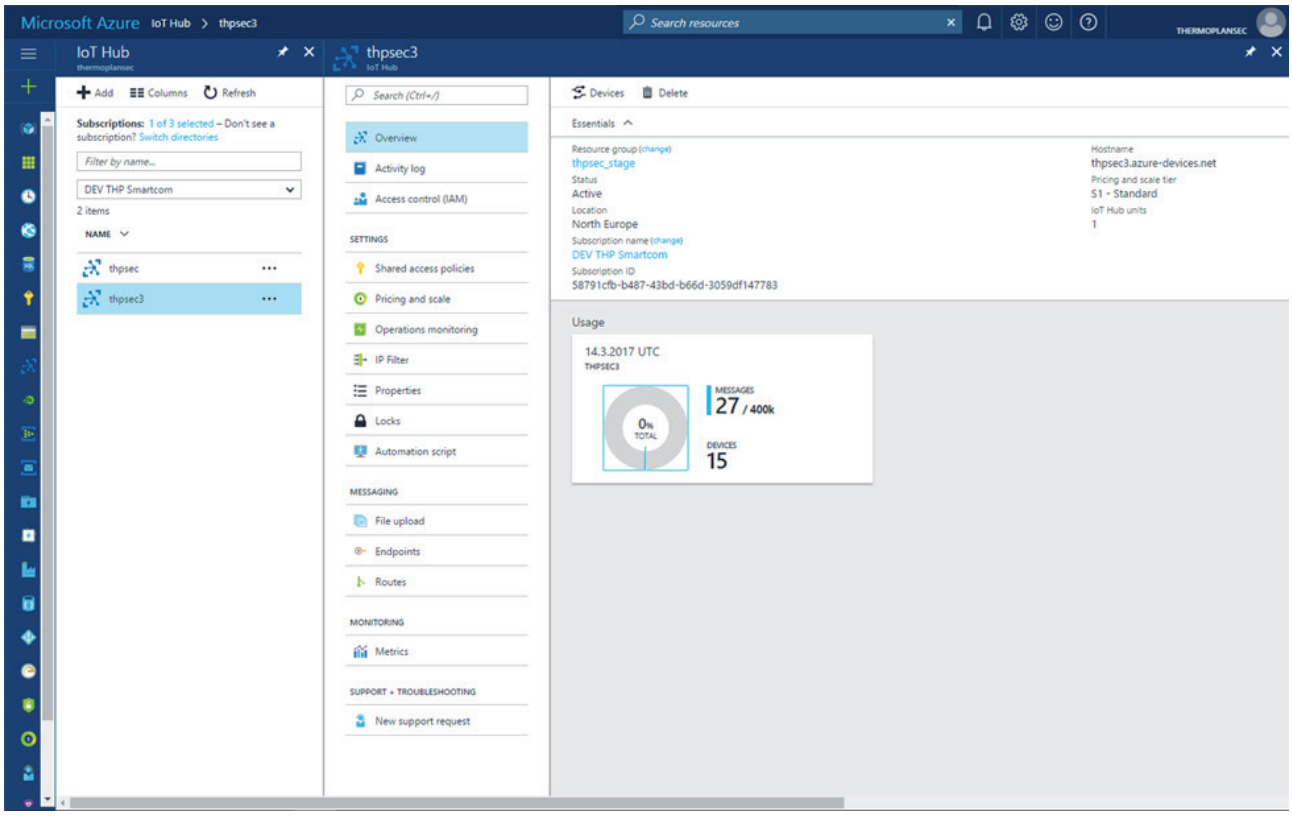
RemoteUpdateStatus

We also have a device-to-cloud message type that reports the progress of an update.

```
    {
    "type": "RemoteUpdateStatus",
    "version": "1.5.0",
    "eventtimelocal": "2017-03-14T18:21",
    "eventtime": "2017-03-14T16:21Z",
    "body": {
    "jobId": 56,
    "status": "failed",
    "message": "another update in progress"
    }
    }
```

With the potential of 50,000 machines each sending up to 1,000 2-KB messages per day, we could have a total of about 50 GB of data per day and up to 1,000 messages per second.

Report showing error messages



SDKs and languages used

Azure IoT device SDK for C

Data processing

After the data is captured by IoT Hub, it is processed by two separate Azure Stream Analytics jobs, each one using a separate view of the message stream through two consumer groups.

The first job takes messages with product production-related information and stores them to Azure Data Lake for persistency and later analysis. Error messages (Error) are sent to a queue connected to an alert system that will send notifications to the right person. Machine information messages (MachineInfo) are also sent to a dedicated queue and are asynchronously processed to find out changes such as exchanged hardware modules due to maintenance.

The second job is responsible for forwarding raw messages to end customers. This is implemented in a multi-tenant fashion so that the messages are sent to the proper owner of the machine. To accomplish this, we implemented a routing table in the form of a CSV file that is fed and used by Azure Stream Analytics to do the correct routing. This could now be achieved directly within IoT Hub thanks to the new message routing feature.

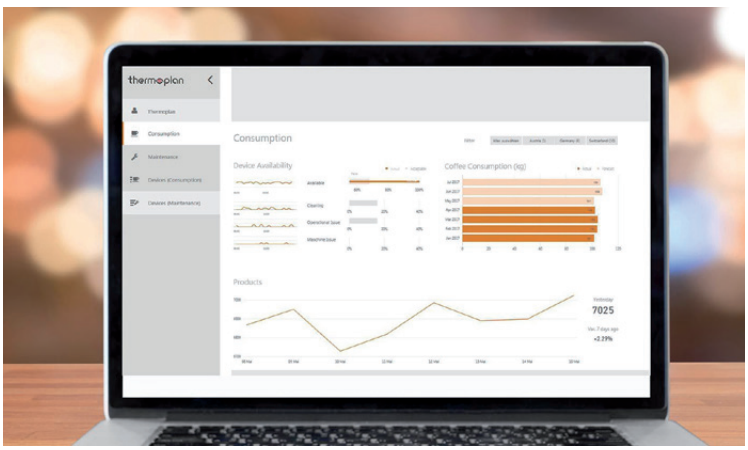
CSV

```
DeviceId,TenantId
00-0000-1,CUSTOMER_X
00-0000-2,CUSTOMER_X
00-0000-3,CUSTOMER_X
00-0000-4,CUSTOMER_X
00-0000-5,CUSTOMER_X
TP0000,CUSTOMER_Y
TP0001,CUSTOMER_Y
TP0002,CUSTOMER_Y
TP0004,CUSTOMER_Y
```

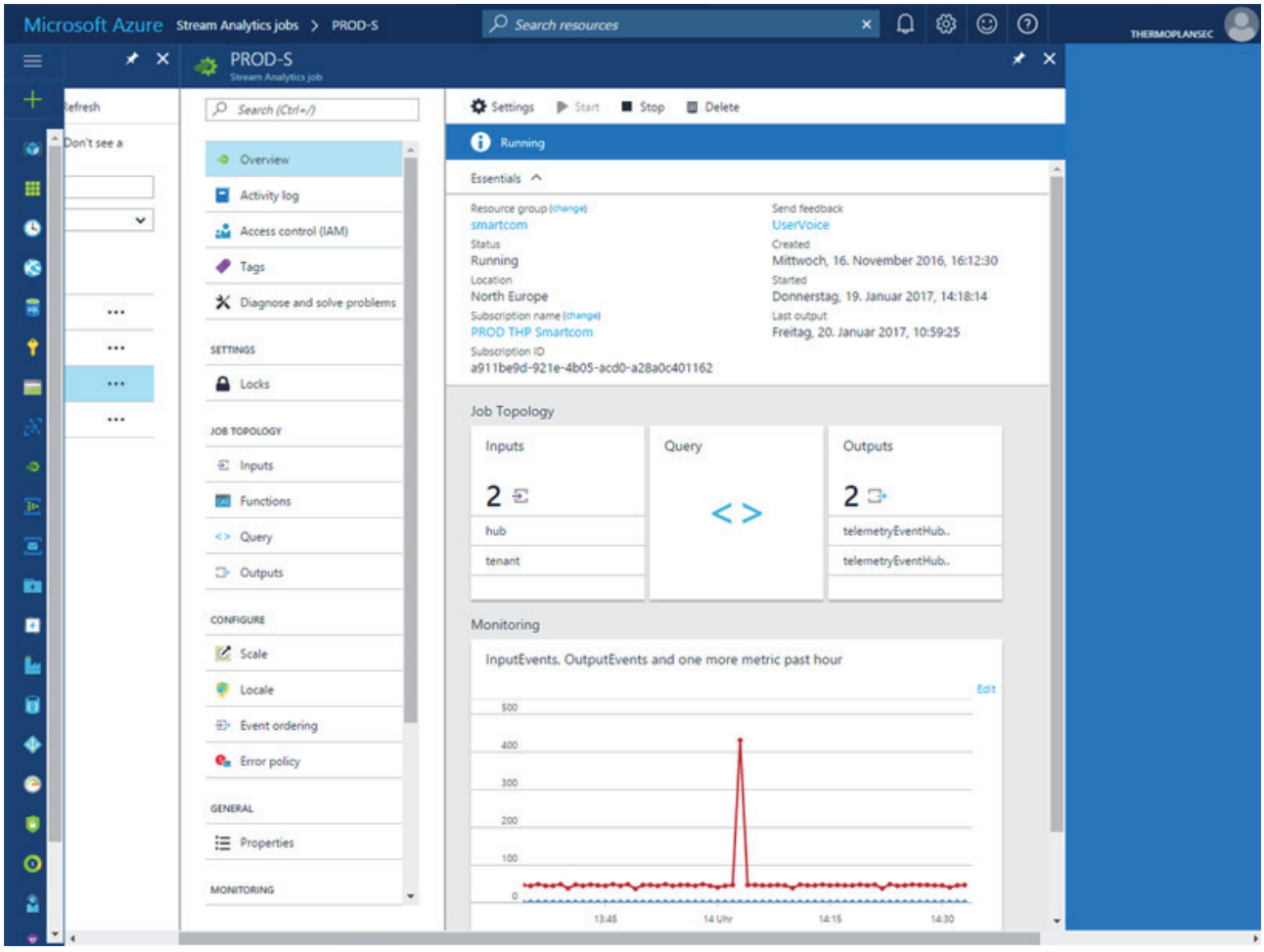
Azure Stream Analytics query sample

```
    WITH AllMessages AS (
            SELECT
                    type,
                    version,
                    eventtime,
                    eventtimelocal,
                    IoTHub.ConnectionDeviceId as deviceId,
                    body
            FROM
                hub PARTITION BY deviceId
            WHERE
            (version = '1.3.0' or version = '1.4.0')
                AND (type = 'Error' or type = 'MachineInfo' or type =
            )

    SELECT
            m.type,
            m.version,
            m.eventtime,
            m.eventtimelocal,
            m.deviceId,
            m.body
    INTO
        telemetryEventHubCUSTOMER_X
    FROM
        AllMessages m PARTITION BY deviceId
    JOIN tenant t ON m.deviceId = t.Deviceid
    WHERE
        t.TenantId = 'CUSTOMER_X'
```



Azure Stream Analytics in action



Learnings from the Microsoft team and the customer team

Due to the large number of messages and their size, the initial idea was to use the more size-optimized binary format Apache Avro. Unfortunately, this has not been possible because a schema is needed for processing a message through Azure Stream Analytics. The best solution would be to have a mechanism that injects the right schema within Stream Analytics, and which doesn't require the Avro message to contain the schema itself. Unfortunately, this is currently not possible, so we would need to provide the schema directly within every single message from the client device. Although this would be technically feasible, in our case it would defeat the goal for a more compact message compared to a pure JSON one.

An alternative would be to batch multiple records within a single message containing the schema, but this does not reflect our goal, which is to send one message every time a coffee is being served.

It would also be possible to do some sort of JSON optimization with short names or types. Considering that we don't have multiple rows of the same type in the message, the gain would not be that big, and it would cost in terms of readability and more complex queries within Stream Analytics to do the field matching. Another approach would be to pick up the messages with our custom code instead of relying on Stream Analytics, but we feel that Stream Analytics is helping us to be quicker and more agile, and in the end we will have a less complex solution to maintain.

For all these reasons, we decided to use JSON as the message format, simplifying the flow at the expense of the size and compactness of the message.

Example of Avro schema

```
    {"namespace": "thermoplan.avro",
     "type": "record",
     "name": "Error",
     "fields": [
                {"name": "number", "type": "int"},
                {"name": "severity",  "type": "int"},
                {"name": "internaltext", "type": ["string", "null"]}
                ]
    }
```

Conclusion

This solution results in a big cost savings for both Thermoplan and its end customers. To illustrate, a software update today requires a technician to travel physically to the location of the coffee machine; you can therefore imagine what this means in terms of cost, especially for remote locations.

The solution also offers the potential for monitoring the production of coffee and other products, not only in terms of numbers, but also in terms of quality; this allows the customer to always serve a consistently high quality product.

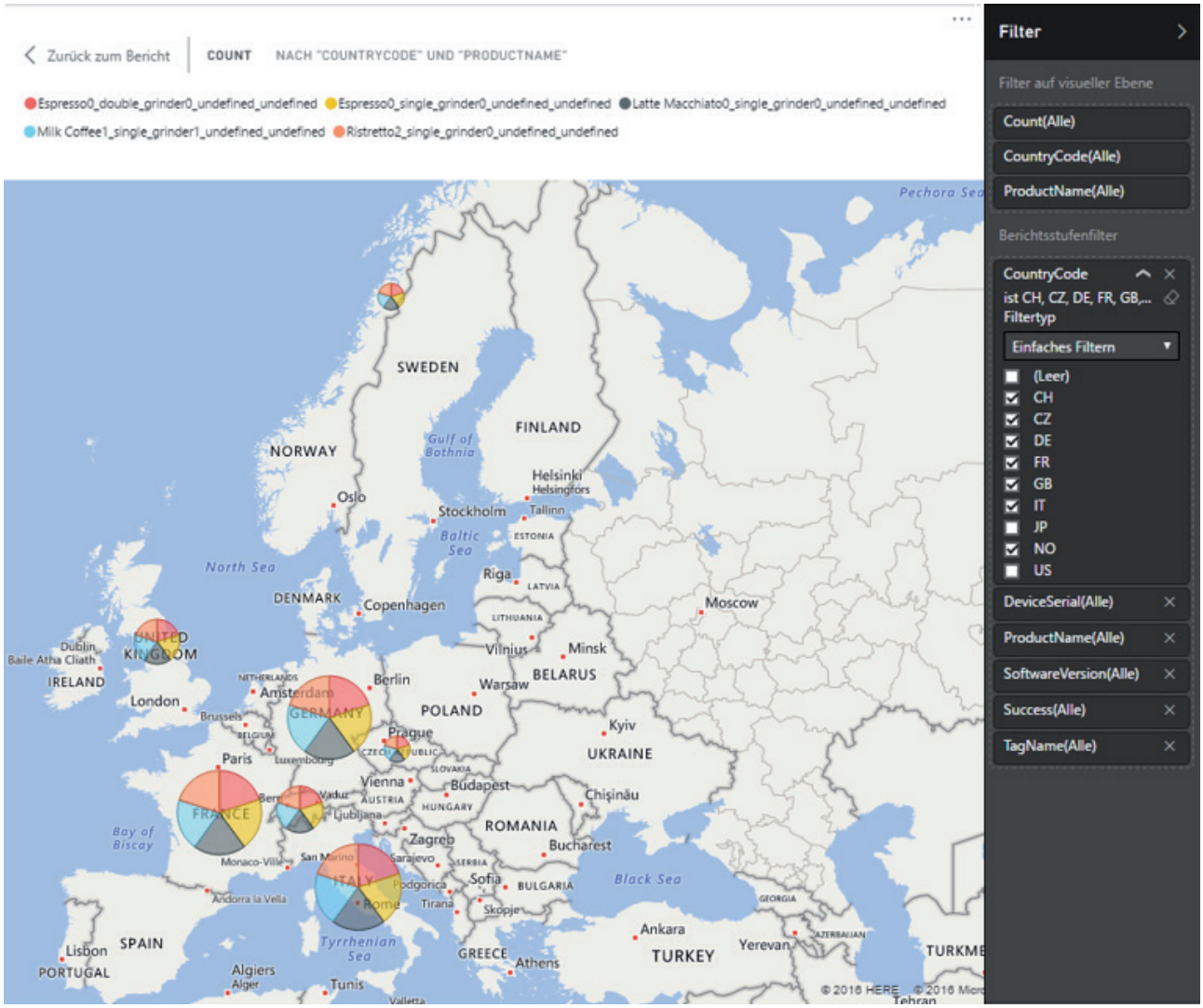
Moreover, thanks to all this telemetry data, Thermoplan is now able to accurately track potential issues and determine the longevity of a machine's various parts. This in the end enables Thermoplan to fulfill the promise of a recognized top quality Swiss product machine.

The plan is to start the production rollout at the end of this summer.

Quote from bbv:
"Microsoft Azure and in particular IoT Hub as a choice has been a very good match that accomplished the requirements of this project. In particular, with a fleet of tens of thousands of coffee machines, IoT Hub met the needed scalability requirements and did so by maintaining cost at a low level. Moreover, having lots of out-of-the-box functionalities provided by services like Azure IoT Hub, Azure Stream Analytics, and Azure Data Lake, it will allow us to have a faster time to market."

Quote from Thermoplan:
"Last but not least, the implementation of a full PaaS solution clearly allowed us to save cost and time in operating the solution compared to a more traditional one. Just to give some numbers, in terms of Azure services we expect the cost per year to stay on a low single-digit amount of US$ per machine per year."

Power BI Embedded report



Opportunities going forward

Thanks to the huge amount of relevant data that will be collected from coffee machines, and predictive technology such as Azure Machine Learning, we expect to be able to better predict maintenance, as well as coffee consumption correlated to location, specific time of year, and season.