OPC AGENT DOCUMENTATION

1. Connecting to the OPC UA Server

To connect to the OPC UA Server you need two components – a **OpcAgent.exe** file and a configuration file to connect it to. The configuration file will be described in the following section, as it is key to operating the OPC Agent.

The OPC Agent will ask you for the configuration file address relative to the **OpcAgent.exe** file, and after you insert the file address and press Enter, you will be receive an information, depending if the configuration file was correct. If the file was incorrect the OpcAgent will prompt you to modify the configuration file, press any key to restart the program and input the file path again.

If the configuration file is read correctly, the resulting output should be similar to this:

```
Reading the config file
Insert the config file path relative to the program
DeviceConfig.xml
Debug value set to "false", all information except for Direct Methods calls will be suppressed
Config file has been successfully loaded
```

After that, we read the nodes of the device on the OPC Server to pass in the D2C message, and allow for calling EmergencyStop and ResetErrorStatus on the device via Direct Methods.

2. OPC Agent configuration

Attached you'll find two files – **DeviceConfigExample.xml** and **ConfigFileSchema.xsd**. Only the XML file is needed for the configuration and startup of the OPC Agent, the ConfigFileSchema file is merely a guideline for how a proper configuration file should look like – if the file gets validated by the schema, it is structurally compatible with the OpcAgent. This is how the DeviceConfigExample file looks like:

```
<DeviceConfig debug="false">
    <ConnectionAddress>OPC Server connection address</ConnectionAddress>
    <AzureConnectionString>Azure IoT Hub connection string</AzureConnectionString>
    <AzureDeviceName>Azure Device Name</AzureDeviceName>
    <ServiceBusConnectionString>Azure Service Bus connection string</ServiceBusConnectionString>
    <RegistryManagerConnectionString>Azure Registry Manager connection string</RegistryManagerConnectionString>
    <EmergencyStopQueueName>Device Errors Service Bus Queue Name</EmergencyStopQueueName>
    <LowerProductionRateQueueName>Lower Production Service Bus Queue Name</LowerProductionRateQueueName>
    <TelemetryDelay>Number of miliseconds (minimum 5000)</TelemetryDelay>
    <Device>
        <Name>Name 1</Name>
    </Device>
    <Device>
        <Name>Name 2</Name>
    </Device>
    <Device>
        <Name>Name 3</Name>
    </Device>
</DeviceConfig>
```

- debug attribute – accepts values "true" and "false" (any other will be defaulted to false), false value will print only crucial information (triggering Direct Methods, small information if there was a problem with receiving messages), while true value will print nearly everything the OpcAgent produces – everything that the false value option provides, but printing telemetry information from the device and providing more information on problems with receiving messages. This element is optional.
- ConnectionAddress –OPC Server connection address
- AzureConnectionString – Azure IoT Hub connection string

- AzureDeviceName – Name of the IoT Device used for sending D2C messages and calling Direct Methods
- ServiceBusConnectionString – Azure Service Bus connection string
- RegistryManagerConnectionString – Azure Registry Manager connection string
- EmergencyStopQueueName – name of the Service Bus Queue used for triggering EmergencyStop on malfunctioning devices
- LowerProductionRateQueueName – name of the Service Bus Queue used for lowering the production rate of devices below 90% KPI
- TelemetryDelay – delay in milliseconds between reading data from the OPC UA Server and passing a D2C message (minimum 5000, if an incorrect value is inserted the program will default to 10000). This element is optional.
- Device and Device/Name – each Device represents a device located on the OPC UA Server, with Name element representing the name of the device.

  It is important to note that each device name is unique – including spaces. While having two separate devices “Device 1” and “Device1” will allow for triggering EmergencyStop and ResetErrorStatus on the correct one, behavior related to ProductionRate and reporting device errors in the Reported Device Twin will be unstable between them.

  For the OPC Agent to function, at least one Device/Name is required.

Below you’ll find an example of a working device config:

```
<DeviceConfig debug="false">
    <ConnectionAddress>opc.tcp://localhost:4840/</ConnectionAddress>
    <AzureConnectionString>HostName=IOTHUBAK2024.azure-devices.net;DeviceId=AKDeviceProjekt;SharedAccessKey=nC6TZc+dPAR/X5gLyU3U4wvMv77tvGUQHQ6q832K/QE=</AzureConnectionString>
    <AzureDeviceName>AKDeviceProjekt</AzureDeviceName>
    <ServiceBusConnectionString>
    Endpoint=sb://servicebusprojektak2024.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=17ex+xaqNWsjLD+w+5bB8WGIYyY2fUdB3+ASbIG8/dg=
    </ServiceBusConnectionString>
    <RegistryManagerConnectionString>HostName=IOTHUBAK2024.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=2vxISIrxXf+rSggeuoL7k9UnPPFogv72vAIoTOT+xvk=
    </RegistryManagerConnectionString>
    <EmergencyStopQueueName>deviceerrorsbus</EmergencyStopQueueName>
    <LowerProductionRateQueueName>kpibus</LowerProductionRateQueueName>
    <TelemetryDelay>15000</TelemetryDelay>
    <Device>
        <Name>Device 1</Name>
    </Device>
    <Device>
        <Name>Device 3</Name>
    </Device>
</DeviceConfig>
```

3. D2C messages

The D2C (Device to Cloud) messages are sent depending on the TelemetryDelay parameter – the minimum value between messages is 5 seconds, while the default value is 10 seconds. There are up to 8 values being transmitted each message:

1. deviceName – name of the physical device, which reports its state to the OPC Agent
2. productionStatus – contains the current production status (stopped = 0, running = 1)
3. workorderID – contains ID of currently running workorder, the value is empty when production is stopped. One workorder ID is processed only by one connected device.
4. goodCount – contains the number of produced good quality items
5. badCount – contains the number of bad quality items
6. temperature – contains the temperature in Celsius
7. deviceErrors – sent in the message only if there was a change on the machine (for example when the machine had an emergency stop triggered)
8. enqueuedTime – DateTime when the message was sent

Following example shows how the message should look like when there was a change in the deviceErrors value (example 1 for “Device 3”) and how it should look like normally (example 2 for “Device 1”).

Sun May 12 2024 13:49:25 GMT+0200 (czas środkowoeuropejski letni):


```
{
  "body": {
    "deviceName": "Device 3",
    "productionStatus": "1",
    "workorderID": "d9776943-4246-45b0-baa0-dc657d805629",
    "goodCount": "98",
    "badCount": "15",
    "temperature": "739",
    "deviceErrors": "12"
  },
  "enqueuedTime": "Sun May 12 2024 13:49:25 GMT+0200 (czas środkowoeuropejski letni)"
}
```

Sun May 12 2024 13:49:24 GMT+0200 (czas środkowoeuropejski letni):


```
{
  "body": {
    "deviceName": "Device 1",
    "productionStatus": "1",
    "workorderID": "e84a7c06-f234-4407-91fc-e95865b80780",
    "goodCount": "59",
    "badCount": "4",
    "temperature": "80,77139509932849"
  },
  "enqueuedTime": "Sun May 12 2024 13:49:24 GMT+0200 (czas środkowoeuropejski letni)"
}
```


4. Device Twin

The Device Twin is our method of both reporting the current state of the machines, as well as ordering the machine to change the value of the production rate.

- Desired Device Twin - If our goal is to lower the production rate of a machine, we need to insert an instruction to the twin via this node Json key-value:
  “**name**_production_rate” : NUMBER

  where:
  - **name** is the Name of the Device reported in Device/Name in the configuration file, containing no spaces (instead of “Device 1_production_rate” insert “Device1_production_rate”)
  - **NUMBER** is a value from 0 to 100, as it represents the production rate measured in %

- Reported Device Twin – each Device in the OPC Agent reports its current production rate and error state in this device twin via this node Json key-value:
  “**name**_production_rate” : NUMBER
  “**name**_error_state” : ERRORNUMBER

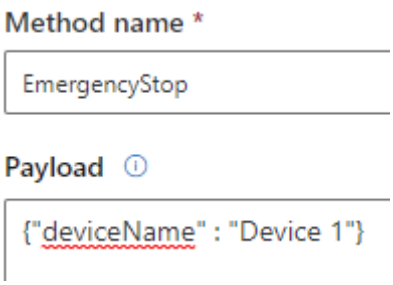
  where:
  - **name** is the Name of the Device reported in Device/Name in the configuration file, containing no spaces (instead of “Device 1_production_rate” insert “Device1_production_rate”)
  - **NUMBER** is a value from 0 to 100, as it represents the production rate measured in %
  - **ERRORNUMBER** is a value from 0 to 15, representing the state of the machine, ranging from no errors (0) to 100% faulty device (15)

```
"properties": {
    "desired": {
        "Device1_production_rate": 80,
        "$metadata": {
            "$lastUpdated": "2024-05-12T12:08:27.0661833Z",
            "$lastUpdatedVersion": 31,
            "Device1_production_rate": {
                "$lastUpdated": "2024-05-12T12:08:27.0661833Z",
                "$lastUpdatedVersion": 31
            }
        },
        "$version": 31
    },
    "reported": {
        "Device1_error_state": "0",
        "Device1_production_rate": "80",
        "Device3_production_rate": "50",
        "Device3_error_state": "12",
        "$metadata": {
            "$lastUpdated": "2024-05-12T12:09:16.9770381Z",
            "Device1_error_state": {
                "$lastUpdated": "2024-05-12T12:09:16.5425041Z"
            },
            "Device1_production_rate": {
                "$lastUpdated": "2024-05-12T12:09:16.5425041Z"
            },
            "Device3_production_rate": {
                "$lastUpdated": "2024-05-12T12:09:16.9770381Z"
            },
            "Device3_error_state": {
                "$lastUpdated": "2024-05-12T12:09:16.9770381Z"
            }
        },
        "$version": 1793
    }
},
```

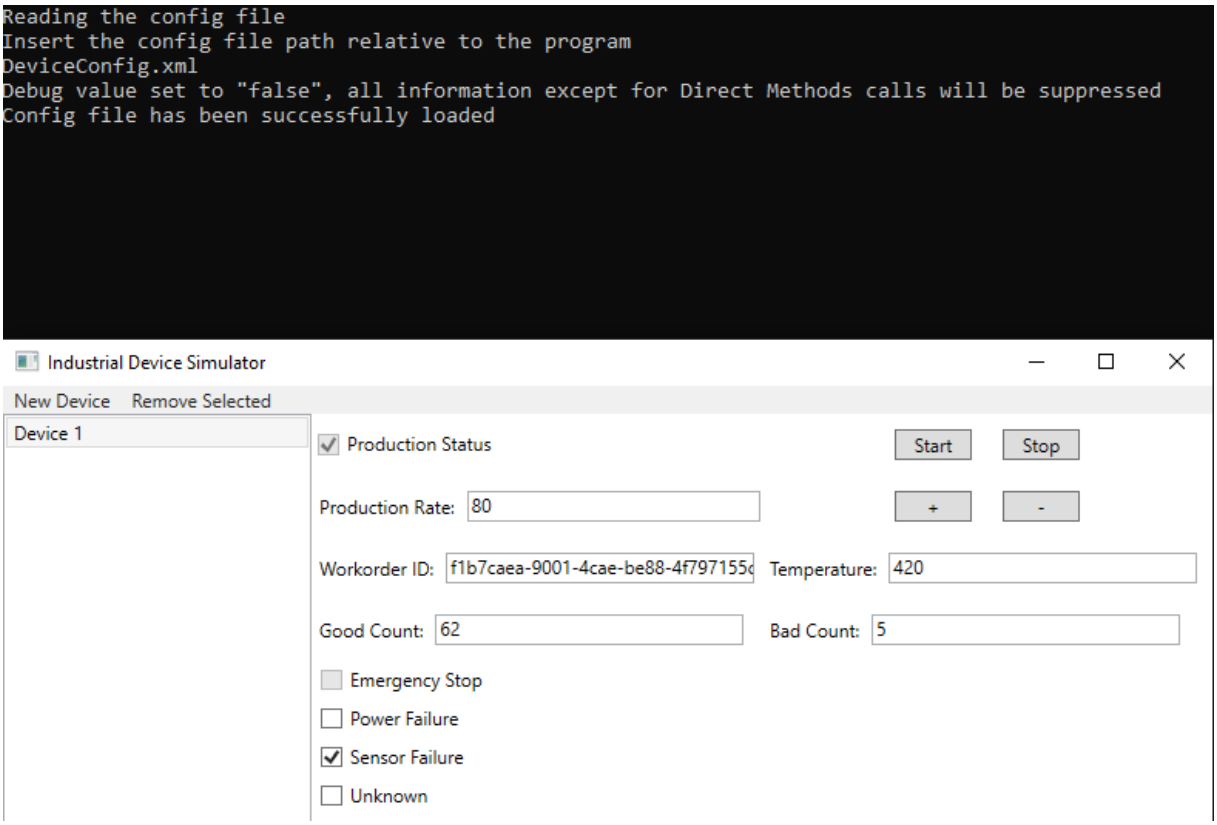

5. Direct Methods

1. Emergency Stop – we call it by name EmergencyStop, and Payload in the format:
   {“deviceName” : “NAME”}
   where NAME = name of the Device on the OPC UA Server we want to call this method for. It is important to point, that unlike the previous section regarding Device Twins, this one needs **exactly** the device name without any alterations. Here is an example of this method in action:

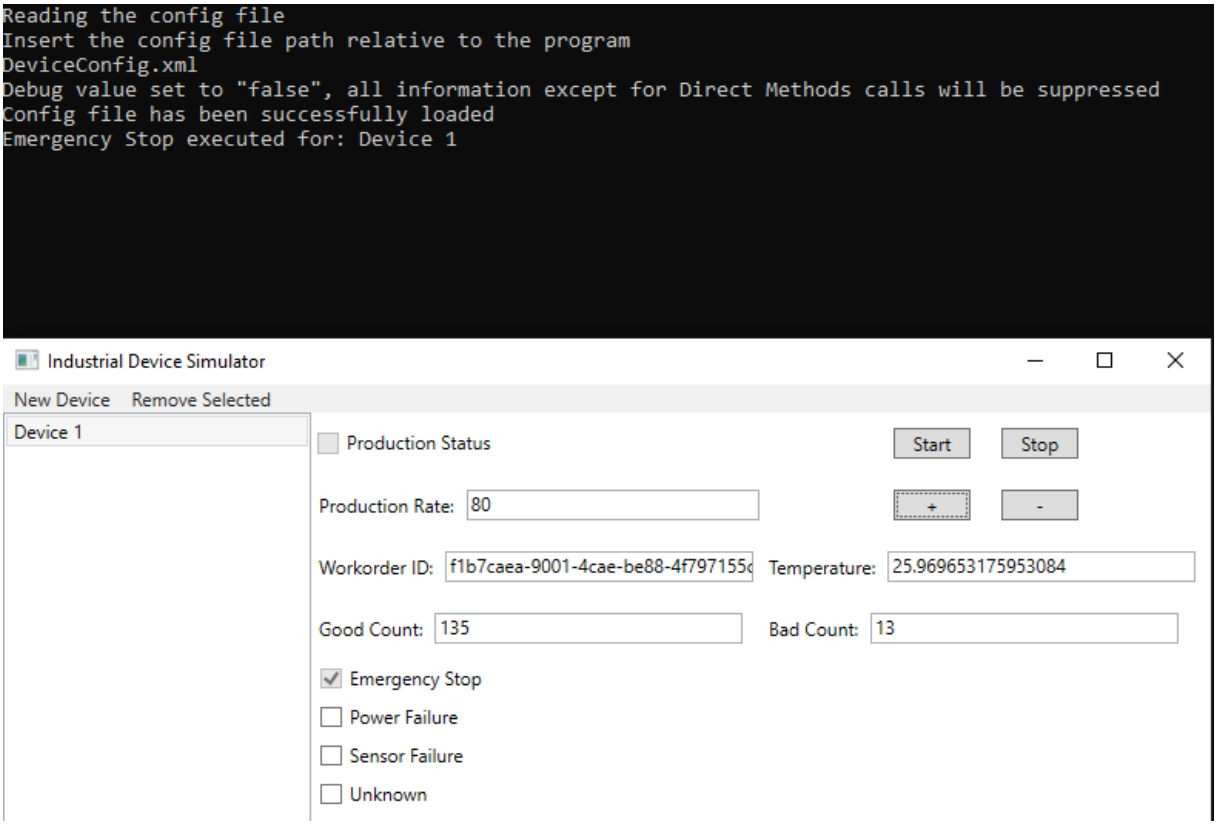
A) How we call it for a device named “Device 1”



B) How the device and the Agent look before executing it:

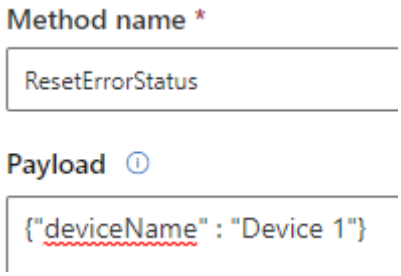


C) And how the device and the Agent look after executing the EmergencyStop method:

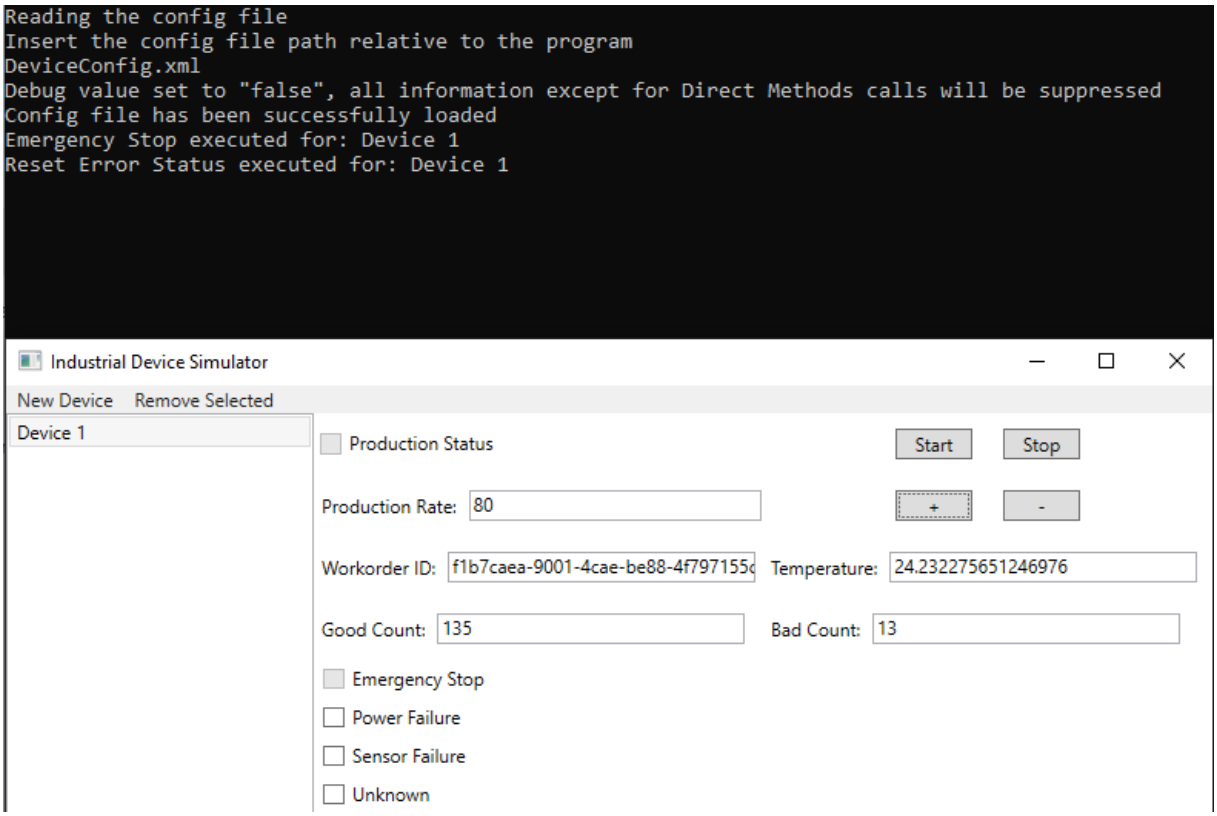


2. ResetErrorStatus – we call it by name ResetErrorStatus , and Payload in the format: {“deviceName” : “NAME”}
   where NAME = name of the Device on the OPC UA Server we want to call this method for. It is important to point, that unlike the previous section regarding Device Twins, this one needs **exactly** the device name without any alterations. Here is an example of this method in action:

A) How we call it for a device named “Device 1”

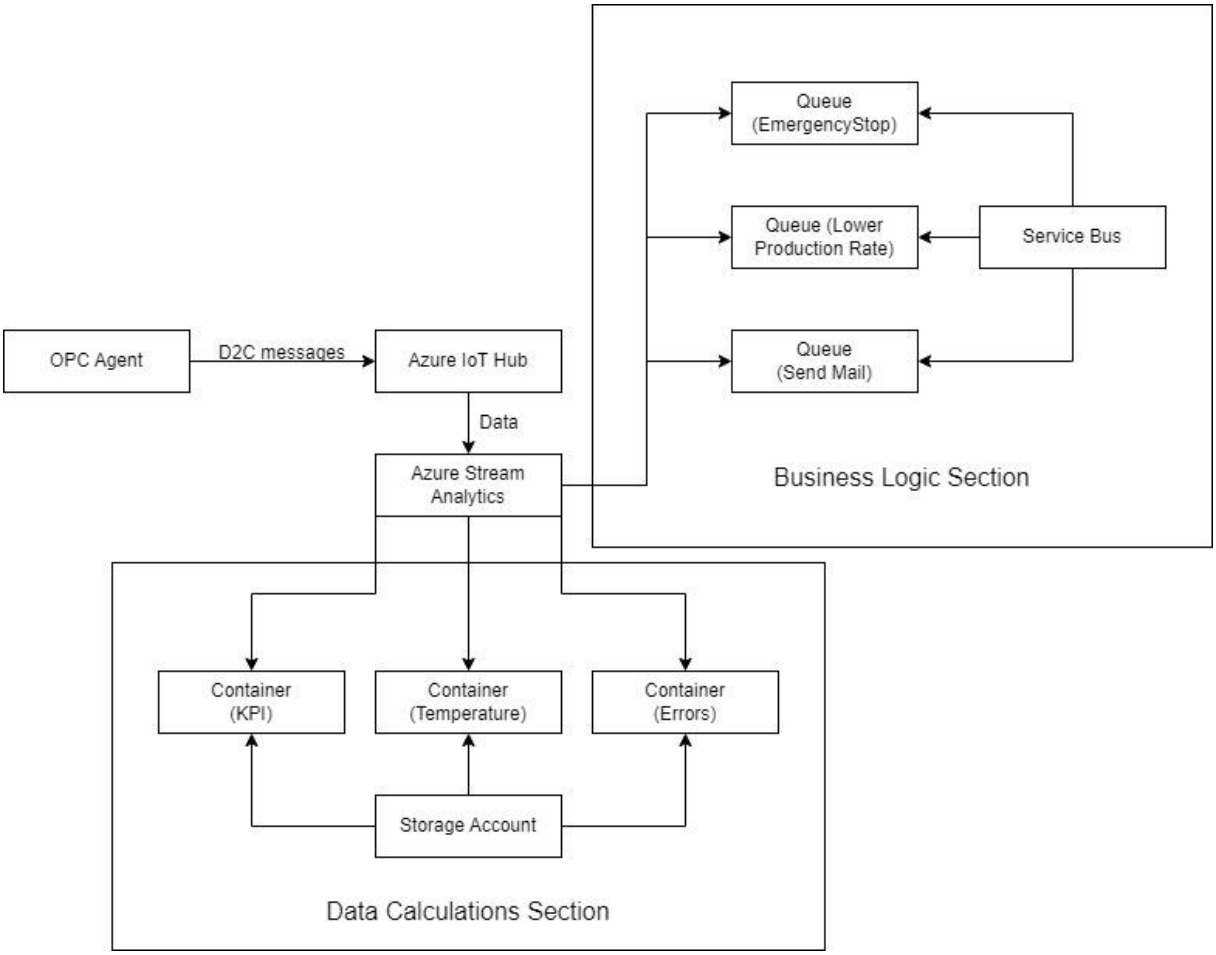


B) And how the device and the Agent look after executing the ResetErrorStatus method (after executing EmergencyStop first, in the previous example):



6. Data calculations and Business Logic

Included with this documentation, you'll find a file **Queries.txt** containing all requested Data Calculations and Business Logic.



All queries are labeled, the input will **always** be the IoT Hub connected to the OPC Agent, while outputs differ for each query. As such, for data calculations:

1. Production KPIs (% of good production (see good count) in total volume, grouped by device in 5-minute windows) are stored in a container referred to as [kpi]
2. Temperature (Every 1 minute average, minimum and maximum temperature over the last 5 minutes is taken and grouped by device) is stored in a container referred to as [temperature]
3. Device errors (Situations whenever a device experience more than 3 errors in under 1 minute) are stored in a container referred to as [deviceerrors]

And for the Business Logic:

1. Emergency Stop request (If a device experiences more than 3 errors in under 1 minute) is sent to a queue referred to as [deviceErrorsBus]
2. Lower Production Rate request (If a device experiences a drop of good production rate below 90% - decrease Desired Production Rate by 10 points) is sent to a queue referred to as [kpibus]
3. Send mail request (If a Device Error of any type occurs) is sent to a queue referred to as [sendMailBus]

It is important to note, that the last Business Logic, Send mail is **NOT** implemented in the OPC Agent – therefore the query connected to it can be omitted from being added into Azure Stream Analytics, as well as the queue in the Service Bus, until this request is implemented on the Agent side.

In short – Data Calculations are stored in separate containers in Storage Account, while requests for action from Business Logic are stored in separate queues in Service Bus. All data comes from the OPC Agent to Azure IoT Hub via a D2C message, which acts as an input for all Stream Analytics queries.

Examples of data in containers:

1. KPI


```
{"deviceName":"Device 1","KPI":84.08,"Datetime of snapshot":"2024-05-11T15:30:00.0000000Z"}
{"deviceName":"Device 3","KPI":61.9,"Datetime of snapshot":"2024-05-11T15:35:00.0000000Z"}
{"deviceName":"Device 1","KPI":85.13,"Datetime of snapshot":"2024-05-11T15:35:00.0000000Z"}
{"deviceName":"Device 3","KPI":62.03,"Datetime of snapshot":"2024-05-11T15:40:00.0000000Z"}
{"deviceName":"Device 1","KPI":83.54,"Datetime of snapshot":"2024-05-11T15:40:00.0000000Z"}
{"deviceName":"Device 3","KPI":70.71,"Datetime of snapshot":"2024-05-11T15:45:00.0000000Z"}
{"deviceName":"Device 1","KPI":84.38,"Datetime of snapshot":"2024-05-11T15:45:00.0000000Z"}
{"deviceName":"Device 3","KPI":75.08,"Datetime of snapshot":"2024-05-11T15:50:00.0000000Z"}
{"deviceName":"Device 1","KPI":86.16,"Datetime of snapshot":"2024-05-11T15:50:00.0000000Z"}
{"deviceName":"Device 1","KPI":86.61,"Datetime of snapshot":"2024-05-11T15:55:00.0000000Z"}
```


2. Temperature


```
{"deviceName":"Device 1","Minimum temperature":-751.0,"Average temperature":-132.05755426342927,"Maximum temperature":416.0,"Datetime of snapshot":"2024-05-07
{"deviceName":"Device 3","Minimum temperature":-718.0,"Average temperature":-96.63051793266175,"Maximum temperature":373.0,"Datetime of snapshot":"2024-05-07T
{"deviceName":"Device 1","Minimum temperature":-751.0,"Average temperature":-81.82611626940603,"Maximum temperature":416.0,"Datetime of snapshot":"2024-05-07T
{"deviceName":"Device 3","Minimum temperature":-718.0,"Average temperature":-55.05836879578481,"Maximum temperature":373.0,"Datetime of snapshot":"2024-05-07T
{"deviceName":"Device 1","Minimum temperature":-751.0,"Average temperature":2.9807873716595212,"Maximum temperature":823.0,"Datetime of snapshot":"2024-05-07T
{"deviceName":"Device 3","Minimum temperature":-718.0,"Average temperature":-26.00897113136227,"Maximum temperature":373.0,"Datetime of snapshot":"2024-05-07T
{"deviceName":"Device 3","Minimum temperature":-473.0,"Average temperature":27.32564698899749,"Maximum temperature":373.0,"Datetime of snapshot":"2024-05-07T1
{"deviceName":"Device 1","Minimum temperature":-751.0,"Average temperature":49.40023038739985,"Maximum temperature":823.0,"Datetime of snapshot":"2024-05-07T1
{"deviceName":"Device 3","Minimum temperature":-473.0,"Average temperature":35.44964463089855,"Maximum temperature":71.78776496579188,"Datetime of snapshot":"
{"deviceName":"Device 1","Minimum temperature":-751.0,"Average temperature":67.66099696198825,"Maximum temperature":823.0,"Datetime of snapshot":"2024-05-07T
```

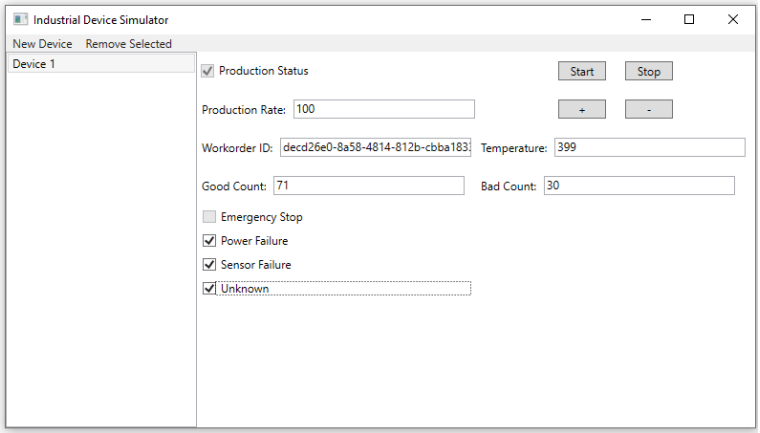

3. Device Errors


```
{"deviceName":"Device 1","Count":4,"Datetime of snapshot":"2024-05-11T15:26:17.3600000Z"}
{"deviceName":"Device 1","Count":4,"Datetime of snapshot":"2024-05-11T15:26:32.9080000Z"}
{"deviceName":"Device 1","Count":4,"Datetime of snapshot":"2024-05-11T15:26:48.5820000Z"}
{"deviceName":"Device 3","Count":4,"Datetime of snapshot":"2024-05-11T15:34:04.9570000Z"}
{"deviceName":"Device 3","Count":4,"Datetime of snapshot":"2024-05-11T15:34:20.9900000Z"}
{"deviceName":"Device 1","Count":4,"Datetime of snapshot":"2024-05-11T15:36:46.0980000Z"}
{"deviceName":"Device 1","Count":4,"Datetime of snapshot":"2024-05-11T15:37:02.1150000Z"}
```


Example of lowered production Rate:

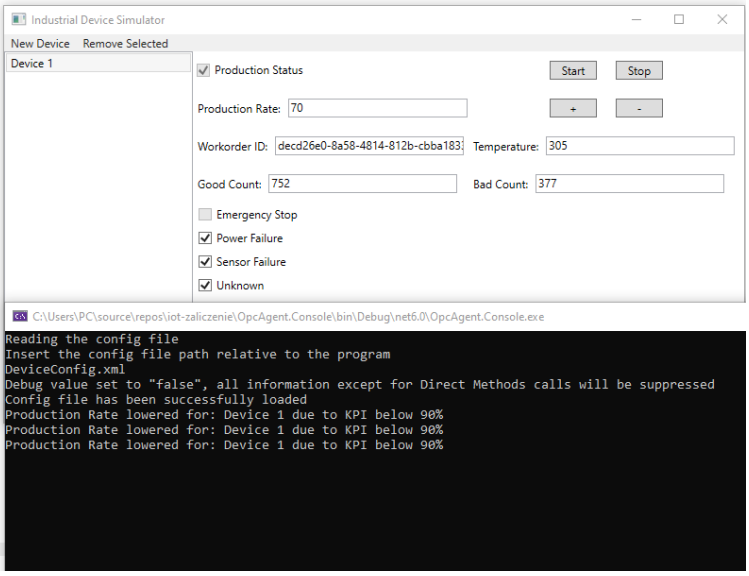
Before adjustment:

```
{
    "deviceId": "AKDeviceProjekt",
    "etag": "AAAAAAAAACA=",
    "deviceEtag": "OTE4Njg2MDM2",
    "status": "enabled",
    "statusUpdateTime": "0001-01-01T00:00:00Z",
    "connectionState": "Connected",
    "lastActivityTime": "2024-05-12T13:36:13.6473269Z",
    "cloudToDeviceMessageCount": 0,
    "authenticationType": "sas",
    "x509Thumbprint": {
        "primaryThumbprint": null,
        "secondaryThumbprint": null
    },
    "modelId": "",
    "version": 2098,
    "properties": {
        "desired": {
            "$metadata": {
                "$lastUpdated": "2024-05-12T14:25:54.9270574Z",
                "$lastUpdatedVersion": 32
            },
            "$version": 32
        },
        "reported": {
            "Device1_error_state": "0",
            "Device1_production_rate": "100",
            "$metadata": {
                "$lastUpdated": "2024-05-12T14:26:20.4303947Z",
                "Device1_error_state": {
                    "$lastUpdated": "2024-05-12T14:26:20.4303947Z"
                },
                "Device1_production_rate": {
                    "$lastUpdated": "2024-05-12T14:26:20.4303947Z"
                }
            },
            "$version": 2066
        }
    },
    "capabilities": {
        "iotEdge": false
    }
}
```

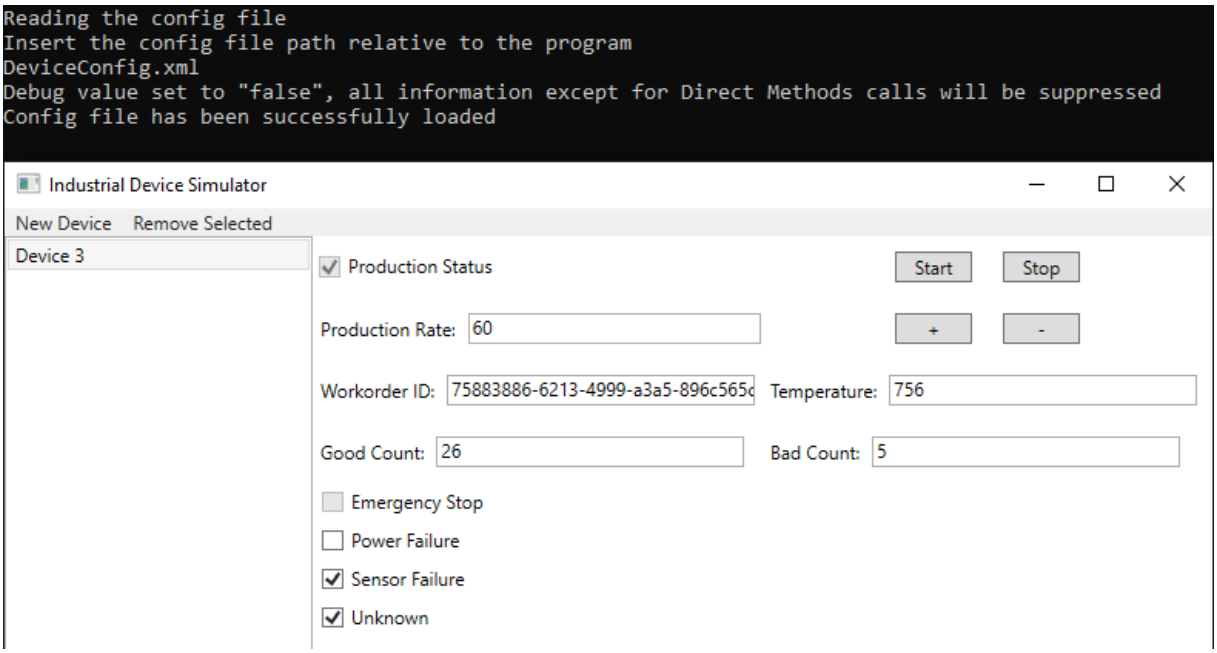
After adjustment:


```
{
    "deviceId": "AKDeviceProjekt",
    "etag": "AAAAAAAAACU=",
    "deviceEtag": "OTE4Njg2MDM2",
    "status": "enabled",
    "statusUpdateTime": "0001-01-01T00:00:00Z",
    "connectionState": "Connected",
    "lastActivityTime": "2024-05-12T14:29:57.2908224Z",
    "cloudToDeviceMessageCount": 0,
    "authenticationType": "sas",
    "x509Thumbprint": {
        "primaryThumbprint": null,
        "secondaryThumbprint": null
    },
    "modelId": "",
    "version": 2125,
    "properties": {
        "desired": {
            "Device1_production_rate": 70,
            "$metadata": {
                "$lastUpdated": "2024-05-12T14:31:32.674067Z",
                "$lastUpdatedVersion": 37,
                "Device1_production_rate": {
                    "$lastUpdated": "2024-05-12T14:31:32.674067Z",
                    "$lastUpdatedVersion": 37
                }
            },
            "$version": 37
        },
        "reported": {
            "Device1_error_state": "14",
            "Device1_production_rate": "70",
            "$metadata": {
                "$lastUpdated": "2024-05-12T14:32:01.3076225Z",
                "Device1_error_state": {
                    "$lastUpdated": "2024-05-12T14:32:01.3076225Z"
                },
                "Device1_production_rate": {
                    "$lastUpdated": "2024-05-12T14:32:01.3076225Z"
                }
            },
            "$version": 2088
        }
    },
    "capabilities": {
        "iotEdge": false
    }
}
```

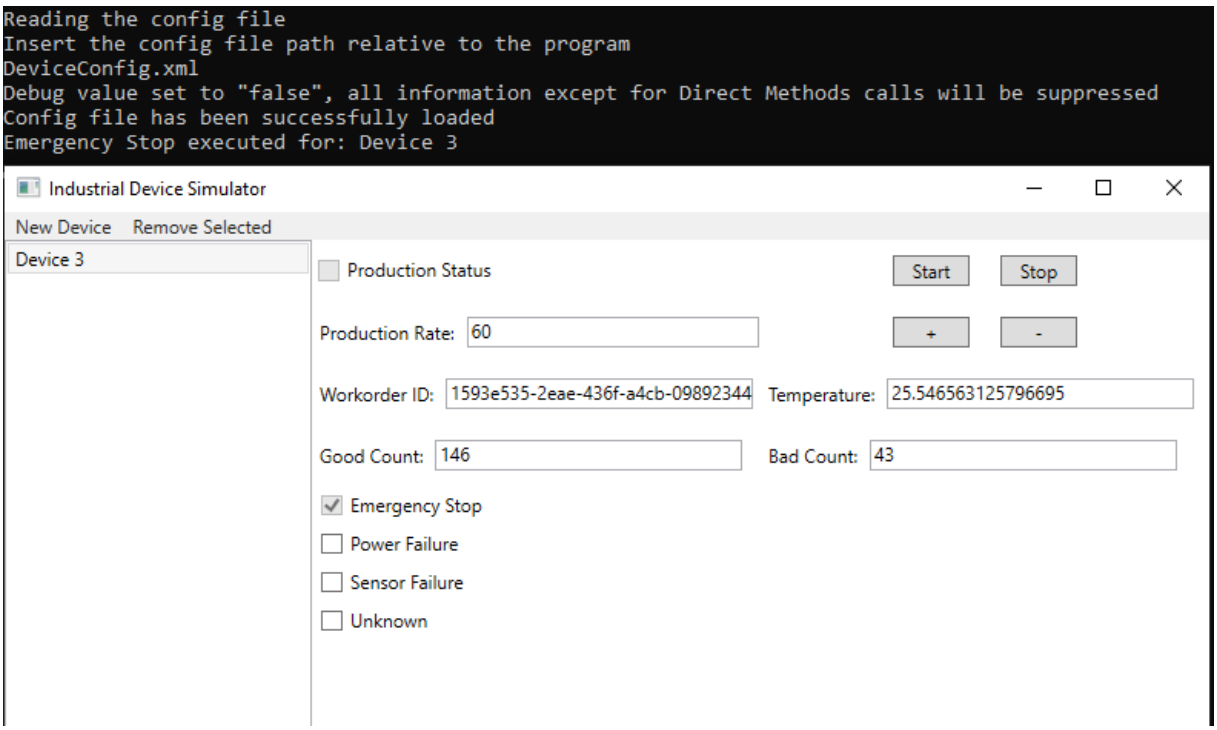



Example of Emergency Stop:

Before the fourth error happened:



After the fourth error happened:



The message in the queue, before being handled by the OpcAgent: