

# Industrial IoT Hands on Lab for IoT Central

<https://github.com/onderyildirim/iotcentral-iiot-lab>

## Prerequisites

- Azure subscription
- Azure CLI with Bash script.
  - Can use Azure Cloud Shell ([shell.azure.com](https://shell.azure.com))
- Git client
- Power BI Desktop
- Azure Data Explorer instance
- (optional) VSCODE

# Objectives using IoT for IIoT



MANAGEMENT AND  
MONITORING OF DEVICES



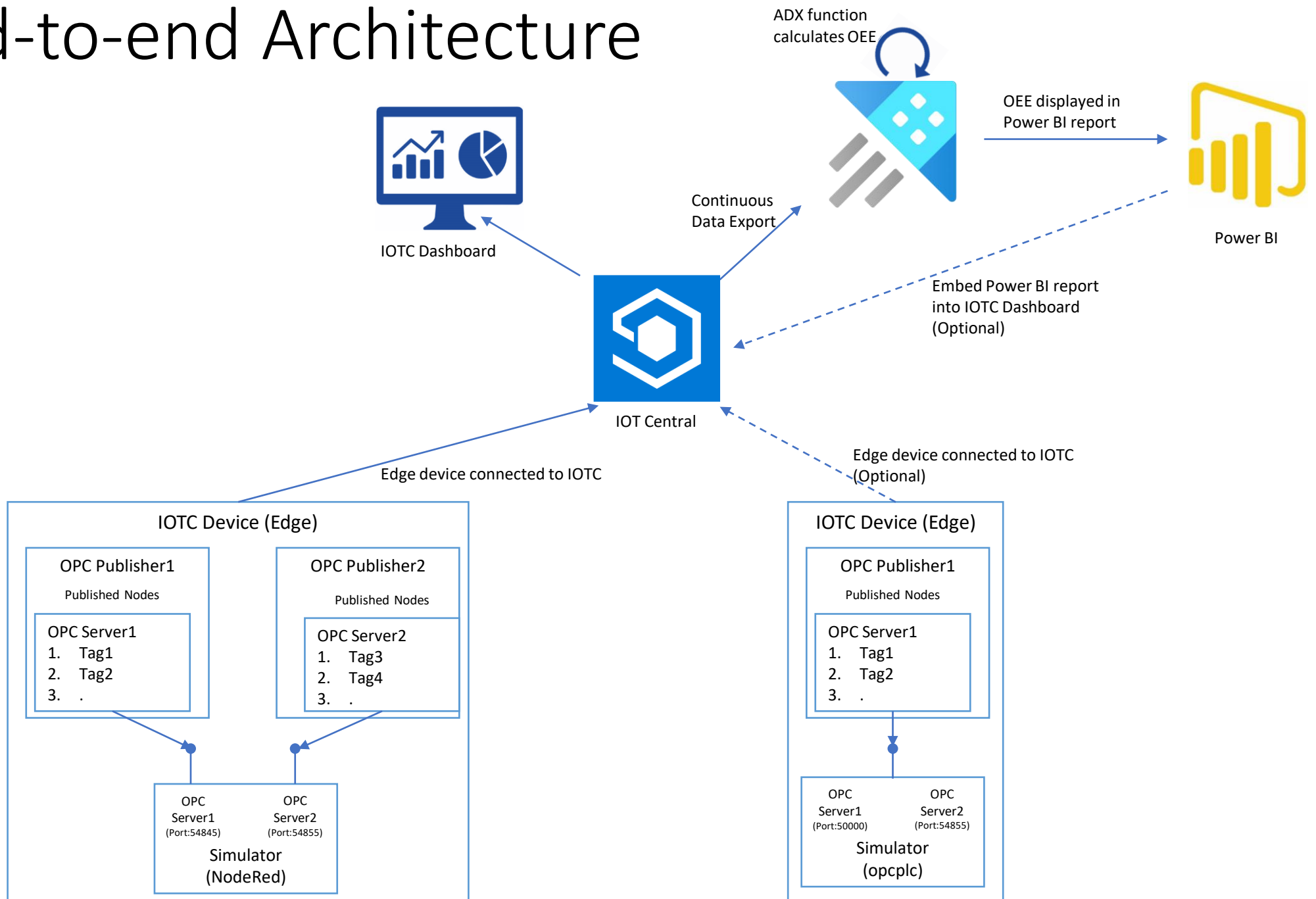
PROVIDING ALERTING  
AND DASHBOARDING



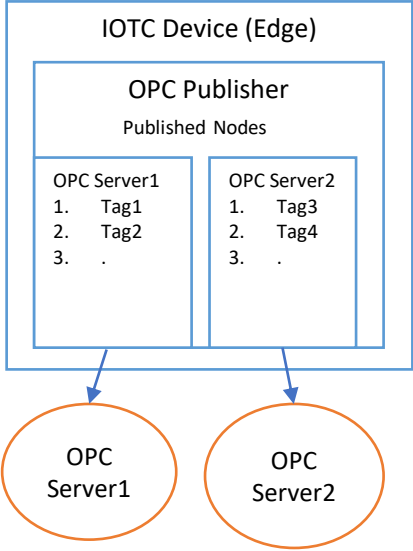
EXPORTING DATA FOR  
FURTHER ANALYSIS

# End-to-end Architecture

4

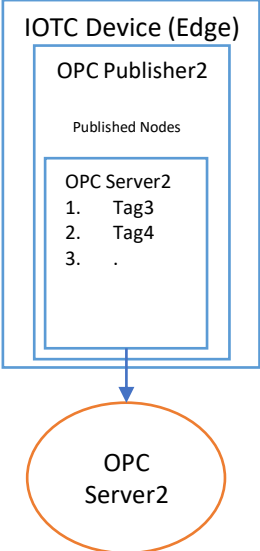


# IOTC-OPC Publisher Possible Configs



- Asset name (source) is maintained in a measure
- Hard to develop IOTC dashboards

```
[
  {
    "EndpointUrl": "opc.tcp://opcuasim:54845/OPCUA/Site1",
    "OpcNodes": [
      {
        "Id": "ns=1;s=STATUS",
      },
      {
        "Id": "ns=1;s=ITEM_COUNT_GOOD",
      },
      {
        "Id": "ns=1;s=ITEM_COUNT_BAD",
      }
    ]
  },
  {
    "EndpointUrl": "opc.tcp://opcuasim:54855/OPCUA/Site2",
    "OpcNodes": [
      {
        "Id": "ns=1;s=STATUS",
      },
      {
        "Id": "ns=1;s=ITEM_COUNT_GOOD",
      },
      {
        "Id": "ns=1;s=ITEM_COUNT_BAD",
      }
    ]
  }
]
}
```



• HOL uses this architecture

terms of data structure  
be used properly  
s properly designed  
edge install/device per asset

# Tips

6

- When you are running scripts, copy the script until the echo command and run.
- Make sure to check the output after each echo command. None of values should be null or empty.
- If your session resets/closes for some reason, use `resetvars.sh` to set global variables back.

# Before we start

7

```
> az --version
```

```
azure-cli                2.37.0
```

```
azure-iot                0.14.0
```

```
kusto                    0.2.0
```

```
az login
```

```
cd ~
```

```
rm -rf ./iotcentral-iiot-lab
```

```
git clone https://github.com/onderyildirim/iotcentral-iiot-lab.git
```

```
cd iotcentral-iiot-lab
```

```
rname="iotchol"
```

```
regionname="eastus2"
```

```
instanceid="$RANDOM"
```

```
adxname="{your ADX name}"
```

```
adxrgname="{your ADX resource group name}"
```

```
appname="iotchol-$instanceid"
```

```
echo "ApplicationName:$appname ResourceGroupName:$rgname RegionName:$regionname  
InstanceId:$instanceid"
```

```
az group create --name $rgname --location $regionname
```

\*Modify **marked** lines according to your environment

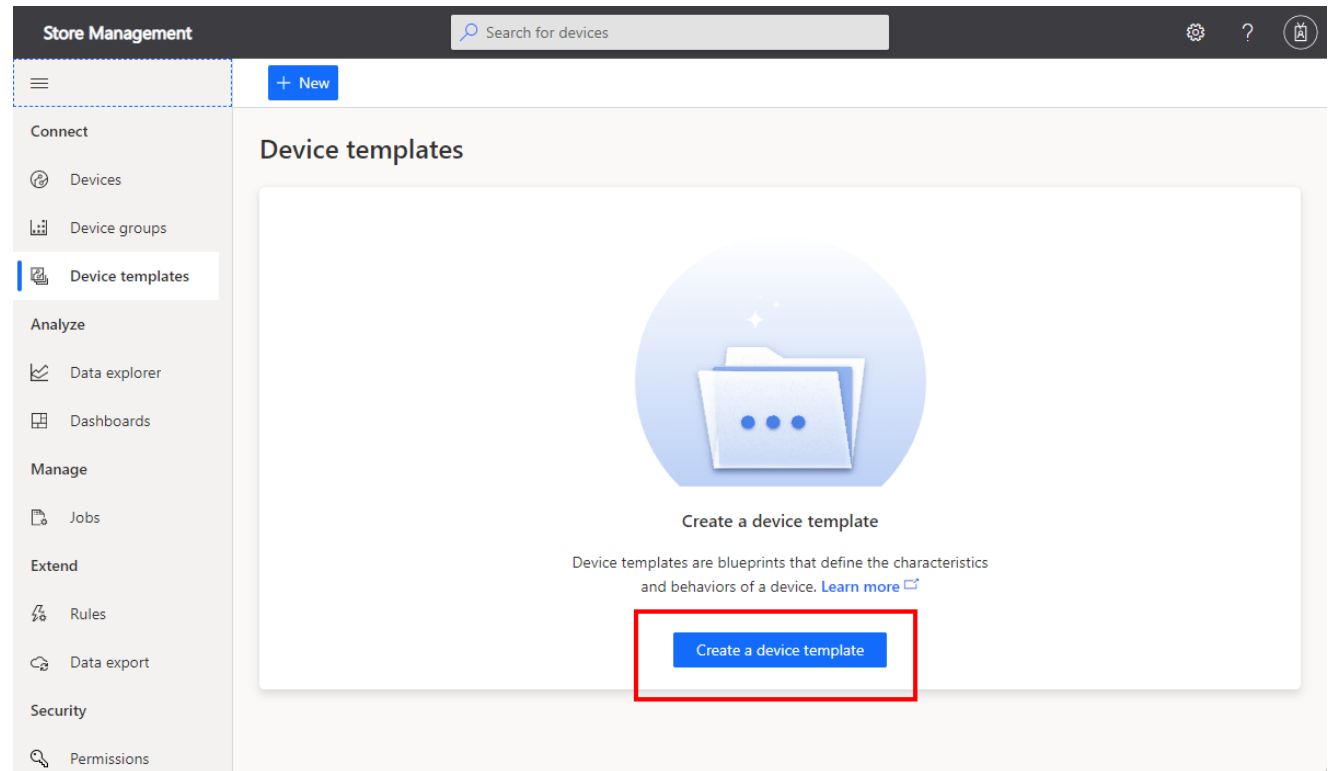
# Create IOTC App

```
appid=$(az iot central app create \  
--resource-group $rgname \  
--name $appname --sku ST0 --location $regionname \  
--subdomain $appname \  
--display-name 'IOTC HOL' --query applicationId --output tsv)  
  
echo "ApplicationId:$appid"  
  
az iot central app identity assign --name $appname --resource-group $rgname --system-  
assigned  
  
echo "You can now navigate to: https://$appname.azureiotcentral.com/device-templates"
```



# Create template for OPCDevice

- Select **Create a device template**,
- Choose the **Azure IoT Edge** tile
- Select **Next: Customize**.
- Enter “**OPCDevice**” as the device template name.
- Select “**Browse**” to upload a deployment manifest.
- Select the “**opcua-edgeDeploymentManifest.json**” file. IoT Central shows **Validated** after it checks the manifest.
- Select **Next: Review**.
- Select **Create** to create the device template.



# Create template for OPCDevice

- Click **“Module asset1”**
- Select **“Edit DTDL”**
- Copy and paste contents from **“opcua1.template.dtdl.json”**

Device templates > OPCDevice > Modules > Module Asset1

OPCDevice  
Application updated: Never Interfaces published: Never

Model  
Azure IoT Edge Capability Model I4mi...

Modules  
Module opcuasim  
**Module Asset1**  
Module asset2  
Cloud properties  
Raw data  
Customize  
Views

Module Asset1 **Root** Draft  
Add capabilities specific to this device model. [Learn more](#)

Save + Add capability Edit identity Version Export Delete ... Edit DTDL

Display name	Name *	Capability type *	Semantic type		
Status	Status	Telemetry	State	×	✓
ItemCountGood	ItemCountGood	Telemetry	None	×	✓
ItemCountBad	ItemCountBad	Telemetry	None	×	✓
Asset	Asset	Telemetry	None	×	✓

+ Add capability

- Click **“Module asset2”**
- Select **“Edit DTDL”**
- Copy and paste contents from **“opcua2.template.dtdl.json”**
- Click **Publish**

Device templates > OPCDevice > Modules > Module Asset2

OPCDevice  
Application updated: Never Interfaces published: Never

Model  
Azure IoT Edge Capability Model I4mi...

Modules  
Module opcuasim  
Module Asset1  
**Module Asset2**  
Cloud properties  
Raw data  
Customize  
Views

Module Asset2 **Root** Draft  
Add capabilities specific to this device model. [Learn more](#)

Save + Add capability Edit identity Version Export Delete ... Edit DTDL

Display name	Name *	Capability type *	Semantic type		
Status	Status	Telemetry	State	×	✓
ItemCountGood	ItemCountGood	Telemetry	None	×	✓
ItemCountBad	ItemCountBad	Telemetry	None	×	✓
Asset	Asset	Telemetry	None	×	✓

+ Add capability

# Create IOT Device for OPCDevice

```
deviceid="opcl"
```

```
devicetemplate="OPCDevice"
```

```
devicetemplateid=$(az iot central device-template list --app-id  
$appid --compact --query "[?displayName.contains(@,  
'$devicetemplate')].\"@id\"" -o tsv))
```

```
echo "DeviceTemplateID=$devicetemplateid "
```

```
az iot central device create --app-id $appid --device-id $deviceid  
--template $devicetemplateid
```

```
idscope=$(az iot central device show-credentials --app-id $appid  
--device-id $deviceid --query "idScope" -o tsv))
```

```
devicekey=$(az iot central device show-credentials --app-id  
$appid --device-id $deviceid --query "symmetricKey.primaryKey" -o  
tsv))
```

```
echo "DeviceId=$deviceid ID Scope=$idscope DeviceKey=$devicekey "
```

# Create network

```
networkName="iotchol-network-$instanceid"
```

```
echo "NetworkName:$networkName"
```

```
az deployment group create --name NetworkDeployment --resource-group "$rgname" --  
template-file "./networkdeploy.json" --parameters networkName="$networkName"
```

# Create VM for OPCDevice

```
opcVM="opcdevicevm-${instanceid}"
adminUserSshPublicKey=$(cat $(readlink -f ~/.ssh/id_rsa.pub))
echo "Creating virtual machine $opcVM"
echo " using key $adminUserSshPublicKey"

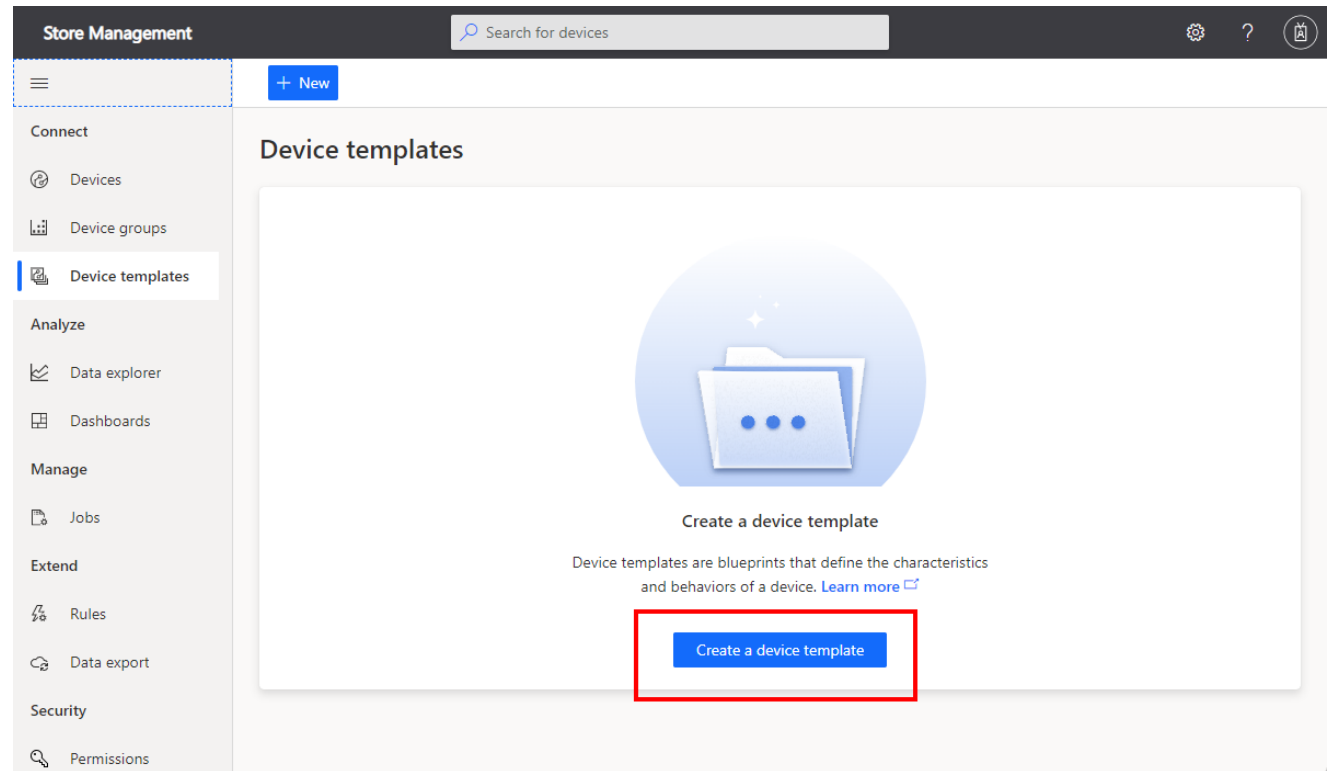
//if SSH public key is empty, you can use following to create one
//ssh-keygen -m PEM -t rsa -b 4096

opcVMDeploymentOutput=$(az deployment group create --name OPCVMDeployment --resource-group "$rgname"
--template-file "opcua-edgeVMTemplate.json" --parameters vmMachineName="$opcVM"
networkName="$networkName" adminUserName="azureuser" adminUserSshPublicKey="$adminUserSshPublicKey"
vmSize="Standard_B1ms" deviceId="$deviceId" scopeId="$idscope" deviceKey="$devicekey" --query
"properties.outputs.[vmMachineName.value, vmMachineFqdn.value, vmAdminUserName.value]" -o tsv))

vmMachineName=${opcVMDeploymentOutput[0]}
vmMachineFqdn=${opcVMDeploymentOutput[1]}
vmAdminUserName=${opcVMDeploymentOutput[2]}
echo "OPC VM SSH : ssh ${vmAdminUserName}@${vmMachineFqdn}"
```

# (Optional) Create device template for PLCDevice

- Select **Create a device template**, choose the **Azure IoT Edge** tile,
- Select **Next: Customize**.
- Enter “**PLCDevice**” as the device template name.
- Select **Browse** to upload a deployment manifest.
- Select the “**opcplc-edgeDeploymentManifest.json**” file. IoT Central shows **Validated** after it checks the manifest.
- Select **Next: Review**.
- Select **Create** to create the device template.



# (Optional) Create device template for PLCDevice

- Click on device template **PLCDevice**
- Click **“Module opcplc”**
- Select **“Edit DTDL”**
- Copy and paste contents from **“opcplc.template.dtdl.json”**
- Click **Publish**

The screenshot shows the 'Module opcplc' configuration page in the Azure IoT Hub Device Template Editor. The left sidebar contains a navigation menu with options: Model, Modules, Module opcplc (selected), Cloud properties, Raw data, Customize, and Views. The main area displays the 'Module opcplc' configuration, which is currently in 'Draft' mode. It includes a 'Save' button, an 'Add capability' button, and an 'Edit identity' button. Below these, there is a table of capabilities. The table has four columns: 'Display name', 'Name \*', 'Capability type \*', and 'Semantic type'. There are seven rows of capabilities, each with a 'Display name', 'Name', 'Capability type' (all set to 'Telemetry'), and 'Semantic type' (all set to 'None'). Each row also has a delete icon (X) and a dropdown arrow. At the bottom of the table, there is an 'Add capability' button. The breadcrumb at the top reads: 'Device templates > PLCDevice > Modules > Module opcplc'.

Display name	Name *	Capability type *	Semantic type		
AlternatingBoolean	AlternatingBoolean	Telemetry	None	X	▼
RandomSignedInt32	RandomSignedInt32	Telemetry	None	X	▼
FastUInt1	FastUInt1	Telemetry	None	X	▼
NegativeTrendData	NegativeTrendData	Telemetry	None	X	▼
PositiveTrendData	PositiveTrendData	Telemetry	None	X	▼
SlowUInt1	SlowUInt1	Telemetry	None	X	▼
SpikeData	SpikeData	Telemetry	None	X	▼

+ Add capability

## (Optional) Create IOT Device for PLCDevice

```
deviceid="plc1"
```

```
devicetemplate="PLCDevice"
```

```
devicetemplateid=$(az iot central device-template list --  
app-id $appid --compact --query "[?displayName.contains(@,  
'$devicetemplate')].\"@id\"" -o tsv))
```

```
echo "DeviceTemplateID=$devicetemplateid"
```

```
az iot central device create --app-id $appid --device-id  
$deviceid --template $devicetemplateid
```

```
idscope=$(az iot central device show-credentials --app-id  
$appid --device-id $deviceid --query "idScope" -o tsv))
```

```
devicekey=$(az iot central device show-credentials --app-id  
$appid --device-id $deviceid --query  
"symmetricKey.primaryKey" -o tsv))
```

```
echo "DeviceId=$deviceid ID Scope=$idscope  
DeviceKey=$devicekey"
```



## (Optional) Create VM for PLCDevice

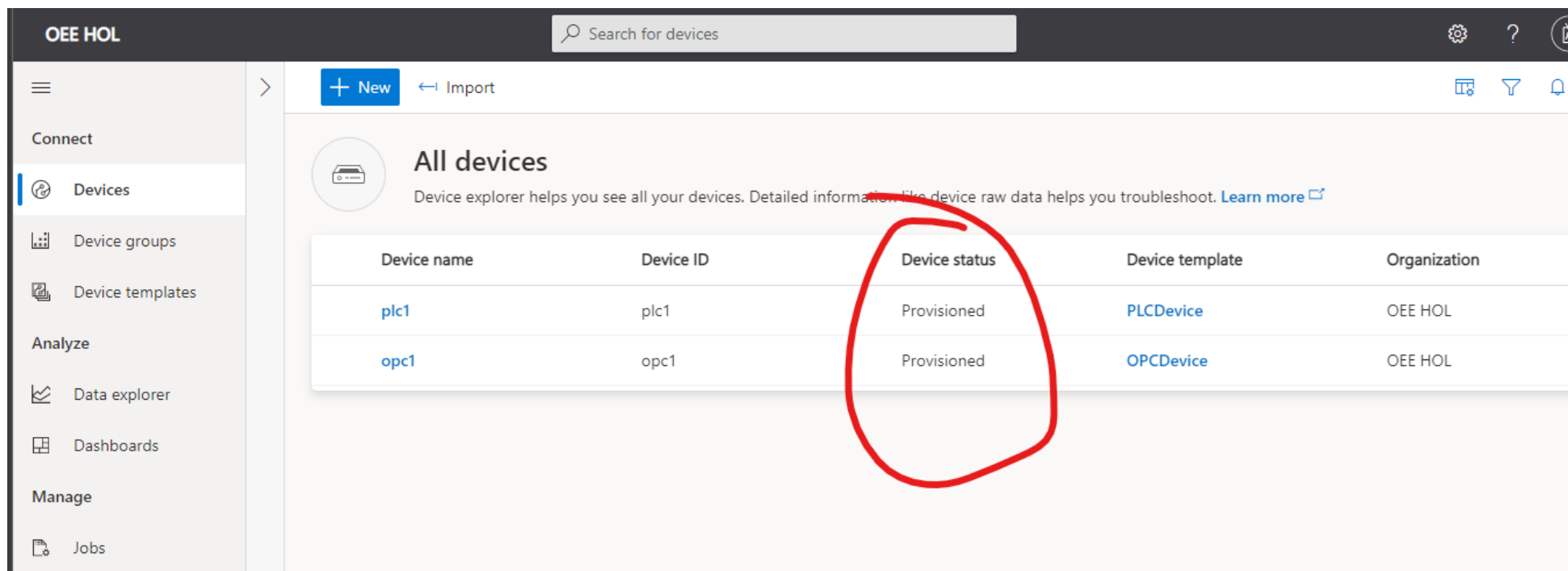
```
plcVM="plcdevicevm-${instanceid}"  
  
echo "Creating virtual machine $plcVM"  
  
echo " using key $adminUserSshPublicKey"
```

```
plcVMDeploymentOutput=($(az deployment group create --name  
PLCVMDeployment --resource-group "$rgname" --template-file "opcplc-  
edgeVMTemplate.json" --parameters vmMachineName="$plcVM"  
networkName="$networkName" adminUserName="azureuser"  
adminUserSshPublicKey="$adminUserSshPublicKey" vmSize="Standard_B1ms"  
deviceId="$deviceId" scopeId="$idscope" deviceKey="$devicekey" --query  
"properties.outputs.[vmMachineName.value, vmMachineFqdn.value,  
vmAdminUserName.value]" -o tsv))
```

```
vmMachineName=${plcVMDeploymentOutput[0]}  
  
vmMachineFqdn=${plcVMDeploymentOutput[1]}  
  
vmAdminUserName=${plcVMDeploymentOutput[2]}  
  
echo "PLC VM SSH : ssh ${vmAdminUserName}@${vmMachineFqdn}"
```

# Check device status

- The **Device status** changes from **Registered** to **Provisioned** when the IoT Edge device connects. May take several minutes.



The screenshot shows the OEE HOL IoT Edge portal. The left sidebar contains navigation links: Connect (Devices, Device groups, Device templates), Analyze (Data explorer, Dashboards), and Manage (Jobs). The main content area is titled 'All devices' and includes a search bar and a table of devices. A red circle highlights the 'Device status' column, which shows 'Provisioned' for both devices.

Device name	Device ID	Device status	Device template	Organization
plc1	plc1	Provisioned	PLCDevice	OEE HOL
opc1	opc1	Provisioned	OPCDevice	OEE HOL

# Check module status

The screenshot shows the Azure IoT Edge portal interface. The left sidebar contains navigation options: Connect, Devices, Device groups, Device templates, Analyze, Data explorer, Dashboards, Manage, Jobs, Extend, Rules, Data export, Security, and Permissions. The main content area displays the status of the device **opc1**, which is connected and has last data received on 6/1/2022 at 4:21:12 PM. The status is Provisioned, and the organization is OEE HOL. The **Modules** tab is selected, showing a summary of 0 stopped modules and 5 modules running. A table lists the modules and their status:

Module Name	Status	Restart Policy	Module Type
<b>\$edgeAgent</b> mcr.microsoft.com/azureiotedge-agent:1.2	Running	Never restarted Restart policy: Always	Azure IoT Edge system modules
<b>\$edgeHub</b> mcr.microsoft.com/azureiotedge-hub:1.2	Running	Never restarted Restart policy: Always	Azure IoT Edge system modules
<b>opcuasim</b> onderyildirim/opcsimulator:0.1.50.0-amd64	Running	Never restarted Restart policy: Always	Custom module
<b>asset1</b> mcr.microsoft.com/iotedge/opc-publisher:2.8	Running	Never restarted Restart policy: Always	Custom module
<b>asset2</b> mcr.microsoft.com/iotedge/opc-publisher:2.8	Running	Never restarted Restart policy: Always	Custom module

# Check raw data

The screenshot shows the OEE HOL interface. The left sidebar contains navigation options: Connect, Devices, Device groups, Device templates, Analyze (Data explorer, Dashboards), and Manage (Jobs). The main area displays the 'opc1' device status as 'Connected' with a green checkmark. Below the status, there are tabs for 'Modules', 'Manage', 'Raw data' (which is highlighted with a red circle), and 'Mapped aliases'. The 'Raw data' tab shows a table with columns: Timestamp, Message type, Event creation time, Module Asset1 / ItemCountBad, and Module Asset1 / ItemCountGood. Two rows of data are visible, both for the timestamp '6/1/2022, 4:22:22 PM' and message type 'Telemetry'. The first row is expanded, showing a JSON payload in a code editor.

```
1 {
2   "_unmodeleddata": {
3     "asset1": {
4       "__t": "m",
5       "MessageId": "647",
6       "MessageType": "ua-data",
7       "PublisherId": "opc.tcp://opcuasim:54845/OPCUA/Site1_71619438",
8       "Messages": [
9         {
10           "DataSetWriterId": "5000",
11           "MetaDataSetVersion": {
```

# Create input mapping for “opc1”

- Go to Devices > **opc1**
- Select **Manage Device > Map Data** From menu
- Fill in mapping as below
- Click **Save**

JSON Path	Alias
<code>\$["Messages"][0]["Payload"]["STATUS"]["Value"]</code>	Status
<code>\$["Messages"][0]["Payload"]["ITEM_COUNT_GOOD"]["Value"]</code>	ItemCountGood
<code>\$["Messages"][0]["Payload"]["ITEM_COUNT_BAD"]["Value"]</code>	ItemCountBad
<code>\$["PublisherId"]</code>	Asset

# Check mapping results for “opc1”

- Go to **Devices > opc1**
- Select **Raw Data**
- Notice mapping in effect, same mapping is applied to all modules (Asset1 and Asset2) and you don't have a way to define them differently for each module
- Also note that Asset1 and Asset2 values come as separate messages

Devices > OPCDevice > opc1

**opc1**  
 Connected | Last data received: 6/2/2022, 1:22:43 PM | Status: Provisioned | Organization: OEE HOL

Modules Manage Raw data Mapped aliases

Timestamp ↓	Message type	Event creation time	Module Asset1 / Asset	Module Asset1 / ItemCountBad	Module Asset1 / ItemCountGo...	Module Asset1 / Sta
① > 6/2/2022, 1:22:43 PM	Telemetry		opc.tcp://opcuasim:54845/O...	6	87	
① > 6/2/2022, 1:22:43 PM	Telemetry		opc.tcp://opcuasim:54845/O...	8	113	
① > 6/2/2022, 1:22:42 PM	Telemetry					
① > 6/2/2022, 1:22:42 PM	Telemetry					
① > 6/2/2022, 1:22:32 PM	Telemetry		opc.tcp://opcuasim:54845/O...	6	97	
① > 6/2/2022, 1:22:32 PM	Telemetry		opc.tcp://opcuasim:54845/O...	2	87	
① > 6/2/2022, 1:22:32 PM	Telemetry					
① > 6/2/2022, 1:22:32 PM	Telemetry					
① > 6/2/2022, 1:22:22 PM	Telemetry		opc.tcp://opcuasim:54845/O...	9	121	

## (Optional) Create input mapping for “plc1”


- Go to **Devices > plc1**
- Select **Manage Device > Map Data** From menu
- Fill in mapping as below
- Click **Save**

JSON Path	Alias
["\$Messages"][0]["Payload"]["SlowUInt1"]["Value"]	SlowUInt1
["\$Messages"][0]["Payload"]["AlternatingBoolean"]["Value"]	AlternatingBoolean
["\$Messages"][0]["Payload"]["RandomSignedInt32"]["Value"]	RandomSignedInt32
["\$Messages"][0]["Payload"]["FastUInt1"]["Value"]	FastUInt1
["\$Messages"][0]["Payload"]["NegativeTrendData"]["Value"]	NegativeTrendData
["\$Messages"][0]["Payload"]["PositiveTrendData"]["Value"]	PositiveTrendData
["\$Messages"][0]["Payload"]["Spikedata"]["Value"]	Spikedata
["\$PublisherId"]	Asset




# (Optional) Check mapping results for “plc1”

- Go to **Devices > plc1**
- Select **Raw Data**
- Notice mapping in effect

Devices > PLCDevice > plc1

 **plc1**  
Connected | Last data received: 6/2/2022, 1:13:28 PM | Status: Provisioned | Organization: OEE HOL

Modules Manage Raw data Mapped aliases

Timestamp ↓	Message type	Event creation time	AlternatingBoolean	FastUInt1	NegativeTrendData	PositiveTrendData	Randon
Create reusable queries for performance insights, and track trends over custom time periods.							
 > 6/2/2022, 1:12:58 PM	Telemetry			597			121566
 > 6/2/2022, 1:12:58 PM	Telemetry			594			835262
 > 6/2/2022, 1:12:48 PM	Telemetry						



# Create dashboard for OPCDevice

- Goto **Dashboards > Edit**
- Drag&Drop a **Line Chart** from left
- Configure **OPCDevice > opc1 > Asset1/ItemCountGood, Asset1/ItemCountBad**
- Drag&Drop **State History** from left
- Configure **OPCDevice > opc1 > Asset1/Status**
- Drag&Drop a **State Chart** from left
- Configure **OPCDevice > opc1 > Asset1/Status**
- Click **Save**

**Configure line chart** ✕

Show legend ⓘ  
☒ On

Show X axis ⓘ  
☒ On

Show Y axis ⓘ  
☒ On

Display range ⓘ  
Past 30 minutes

Interval ⓘ  
1 Minute

Organization ⓘ  
OEE HOL

Device group \* ⓘ  
OPCDevice - All devices

Device Count: 1

Devices \* ⓘ  
opc1

1 selected

▼ Telemetry

Module Asset1 / It... Average

Module Asset1 / It... Average

Update Cancel

**Configure state** ✕

Title \* ⓘ  
State history

Time range ⓘ  
Past 30 minutes

Organization ⓘ  
OEE HOL

Device group \* ⓘ  
OPCDevice - All devices

Device Count: 1

Devices \* ⓘ  
opc1

1 selected

▼ State

Module Asset1 / Status ✕

+ Capability

▼ Tile Format

Text size  
10.5 pt

Decimals ⓘ

Abbreviate values ⓘ  
☐ Off

Wrap text ⓘ

Update Cancel

**Configure state chart** ✕

Title \* ⓘ  
State chart ✕

Time range ⓘ  
Past 30 minutes

Organization ⓘ  
OEE HOL

Device group \* ⓘ  
OPCDevice - All devices

Device Count: 1

Devices \* ⓘ  
opc1

1 selected

▼ State

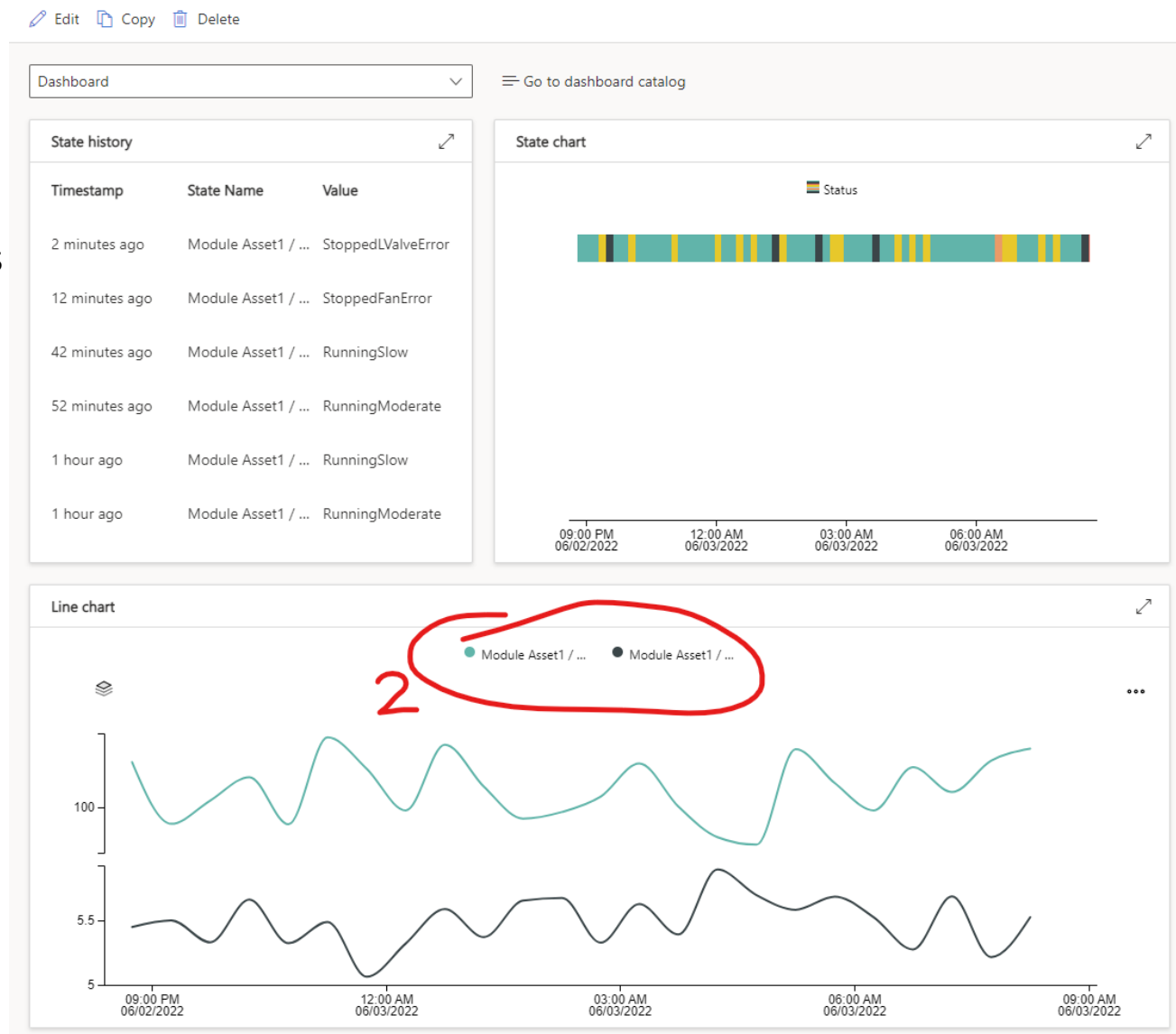
Module Asset1 / Status ✕

+ Capability

Update Cancel

# Dashboard for OPCDevice

- **State History** and **State chart** are useful visualizations for IIOT
- Labels don't work well with multiple modules having measures with same name



# (Optional) Create dashboard for PLCDevice

- Goto **Dashboards > Dashboard Catalog**
- Click **New**
  - Dashboard Name: plc1 Dashboard
  - Dashboard Type: Organizational
- Click **Create**
- Click **Edit**
- Drag&Drop a **Line Chart** from left
- Configure **PLCDevice > plc1 > SlowUInt1, FastUInt1, AlternatingBoolean, NegativeTrendData ...**
- Click **Save**

×

Configure line chart

Title \* ⓘ

Line chart

Show legend ⓘ

On

Show X axis ⓘ

On

Show Y axis ⓘ

On

Display range ⓘ

Past 30 minutes

Interval ⓘ

1 Minute

Organization ⓘ

OEE HOL

Device group \*

PLCDevice - All devices

Device Count: 1

Devices \* ⓘ

plc1

1 selected

Telemetry

SlowUInt1

Average

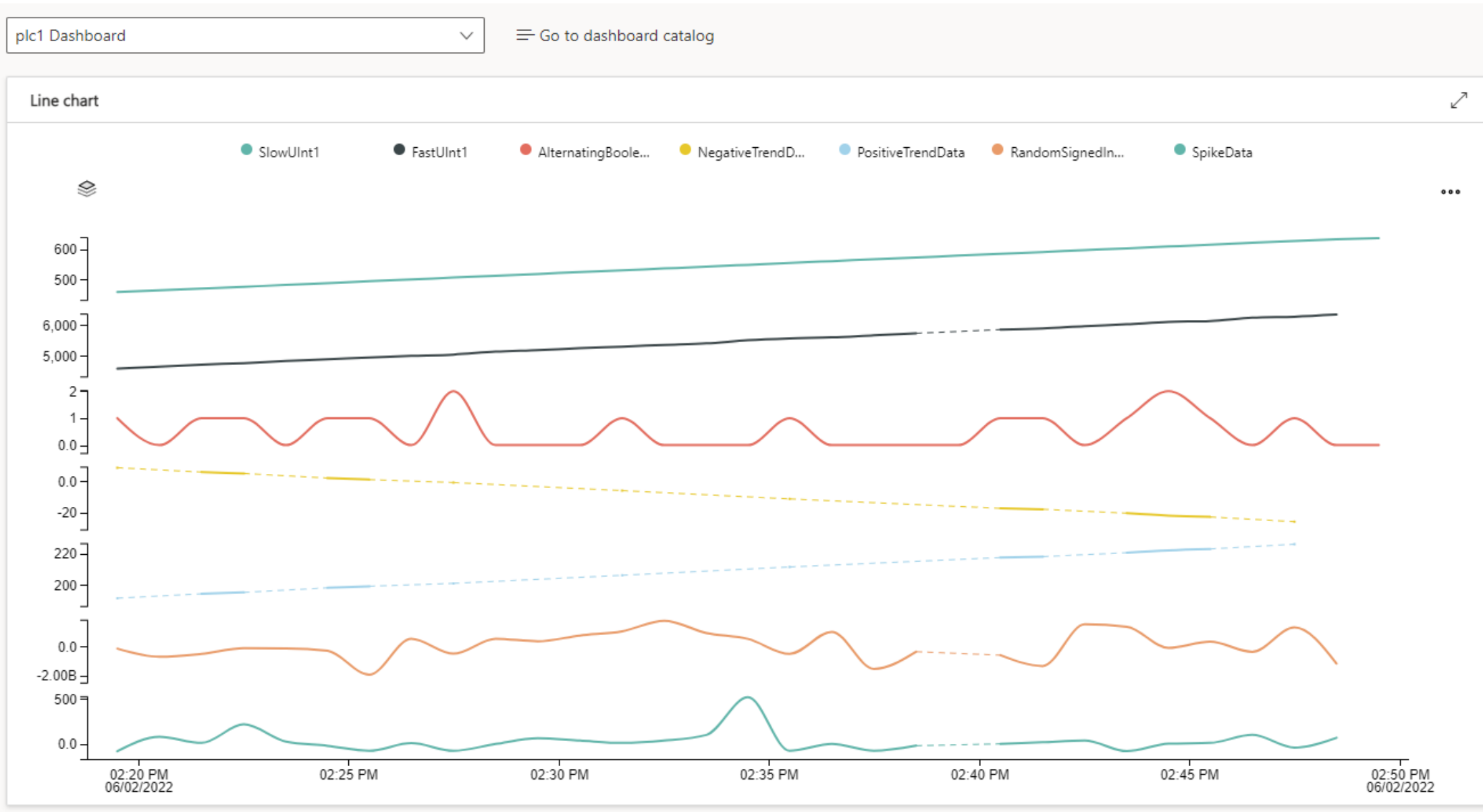
FastUInt1

Average

Update

Cancel

# (Optional) Dashboard for PLCDevice



# Configure ADX for OPCDevice Template

- Create/start an ADX cluster
- Enable streaming ingestion (Azure Portal > ADX Cluster > Configurations > Streaming Ingestion)
- Create a database (iotcholdb)
- Grant IOT Central App ADX access (<https://docs.microsoft.com/en-us/azure/iot-central/core/howto-export-to-azure-data-explorer?tabs=service-principal%2Cjavascript>)
 

```
identityresult=$(az iot central app identity show --name "$appname" --query "[principalId,tenantId]" -o tsv)
principalId=${identityresult[0]}
tenantId=${identityresult[1]}
echo "principalId=$principalId"
echo "TenantId=$tenantId"
az kusto database-principal-assignment create --cluster-name "$adxname" \
  --resource-group "$adxrgname" \
  --database-name "iotcholdb" \
  --principal-assignment-name $appname \
  --principal-id "$principalId" \
  --principal-type App --role Admin \
  --tenant-id "$tenantId"
```
- Goto Azure Portal > ADX Cluster > Query
- Create table schema in your database by running code in **"opcua-kusto-schema.kql"**.
  - **iiotdata**: IoT data from OPCDevice template devices in IOTC
  - **AssetInventory**: Stores asset metadata, specifically capacity of the asset to be used in OEE calculation
  - **ShiftSchedule**: Stores shift start and end times. In HOL, each day is split into 3 shifts of 8h each.
  - Need to run each command that starts with a dot "." separately
- Create function in your database by running code in **"opcua-assetperformance.kql"**
  - Need to run each command that starts with a dot "." separately

## (Optional) Configure ADX for PLCDevice Template

- Goto **Azure Portal > ADX Cluster > Query**
- Create table schema in your database by running code in “**opcplc-kusto-schema.kql**”.
  - `iiotdataplc`: IoT data from PLCDevice template devices in IOTC
  - Need to run each command that starts with a dot “.” separately

# Configure Data Export for OPCDevice – Create Destination

- Goto **Data Export > Destinations > Add Destination**
  - Destination Name: **ADX**
  - ADX Cluster URL: **https://{adxname}.eastus2.kusto.windows.net**
  - Database Name: **iotcholdb**
  - Table Name: **iiotdata**
  - Authorization: **System-assigned Managed Identity**
- Click **Save**

Save Cancel Rename

Destinations > ADX

ADX

To create a new destination, select a destination type and enter the connection information. [Learn more](#)

Destination type \*

Azure Data Explorer

Cluster URL \*

https://{redacted}.eastus2.kusto.windows.net

Database name \*

iotcholdb

Table name \*

iiotdata

Authorization \*

System-assigned managed identity

Configure the following fields associated with the managed identity.  
[Learn more](#)

# Configure Data Export for OPCDevice – Create Export

- Goto **Data Export > Exports > Add Export**
  - Name of export: **ADX Export**
  - Type of data to export: **Telemetry**
  - Destination: **ADX**
- Click **+Filter**
- Select
  - Name=**Device Template Name**
  - Operator=**Equals**
  - Value=**OPCDevice**
- Click **Transform**
- Paste contents of “**opcua-tx-query.txt**” into “**2. Build transformation query**”
- Click **Save**
- Wait for **Export Status** to be “**Healthy**”

Exports > ADXExport

## ADXExport

☒ Enabled

---

### Data

All of your devices will export data unless you add filters to narrow things down. [Learn more](#)

Type of data to export

Telemetry

Export the data if all of the conditions are true

Name \* Operator \*

Device template name Equals

Value \*

OPCDevice

[+ Filter](#) [+ Message property filter](#)

---

### Enrichments

Add additional information to your export. This will appear as a key value pair in exported messages. [Learn more](#)

[+ Custom string](#) [+ Property](#)

---

### Destinations

Select destinations for your export. If you can't find your destination, [create a new one](#).

✓ Data transformation has been added for exporting to ADX. Please save this export to save these changes.

Destination *	Data transformation	Export status	Details
ADX	<a href="#">Edit</a>	✓ Healthy	
<a href="#">+ Destination</a>			



# Configure Data Export for OPCDevice – Verify Export

- Goto **Azure Portal > ADX Cluster > Query**
- Run following to ensure data coming into ADX.
  - Make sure you have two different Asset values (\*Site1\* and \*Site2\*)

iiotdata

	EnqueuedTime	GatewayId	Asset	Status	ItemCountGood	ItemCountBad	applicationId	component	enrichments
>	2022-06-02 20:03...	opc1	opc.tcp://opcuasim:54845/OPCUA/Site1_89E32A7B		90	1	26cb1c6c-809d-4081-910d-2e06a0c81f2d	null	{}
>	2022-06-02 20:03...	opc1	opc.tcp://opcuasim:54845/OPCUA/Site1_89E32A7B		117	5	26cb1c6c-809d-4081-910d-2e06a0c81f2d	null	{}
>	2022-06-02 20:04...	opc1	opc.tcp://opcuasim:54845/OPCUA/Site1_89E32A7B			6	26cb1c6c-809d-4081-910d-2e06a0c81f2d	null	{}
>	2022-06-02 20:04...	opc1	opc.tcp://opcuasim:54845/OPCUA/Site1_89E32A7B		109	4	26cb1c6c-809d-4081-910d-2e06a0c81f2d	null	{}
>	2022-06-02 20:04...	opc1	opc.tcp://opcuasim:54845/OPCUA/Site1_89E32A7B		119	1	26cb1c6c-809d-4081-910d-2e06a0c81f2d	null	{}
>	2022-06-02 20:04...	opc1	opc.tcp://opcuasim:54845/OPCUA/Site1_89E32A7B		89	8	26cb1c6c-809d-4081-910d-2e06a0c81f2d	null	{}
>	2022-06-02 20:04...	opc1	opc.tcp://opcuasim:54845/OPCUA/Site1_89E32A7B		83	4	26cb1c6c-809d-4081-910d-2e06a0c81f2d	null	{}
>	2022-06-02 20:04...	opc1	opc.tcp://opcuasim:54845/OPCUA/Site1_89E32A7B		119	6	26cb1c6c-809d-4081-910d-2e06a0c81f2d	null	{}
>	2022-06-02 20:04...	opc1	opc.tcp://opcuasim:54845/OPCUA/Site1_89E32A7B		108	1	26cb1c6c-809d-4081-910d-2e06a0c81f2d	null	{}
>	2022-06-02 20:04...	opc1	opc.tcp://opcuasim:54845/OPCUA/Site1_89E32A7B		83	6	26cb1c6c-809d-4081-910d-2e06a0c81f2d	null	{}
>	2022-06-02 20:04...	opc1	opc.tcp://opcuasim:54845/OPCUA/Site1_89E32A7B		103	2	26cb1c6c-809d-4081-910d-2e06a0c81f2d	null	{}
>	2022-06-02 20:04...	opc1	opc.tcp://opcuasim:54845/OPCUA/Site1_89E32A7B		97	8	26cb1c6c-809d-4081-910d-2e06a0c81f2d	null	{}
>	2022-06-02 20:04...	opc1	opc.tcp://opcuasim:54845/OPCUA/Site1_89E32A7B		110	1	26cb1c6c-809d-4081-910d-2e06a0c81f2d	null	{}

# Populate Helper Tables in ADX

- Goto **Azure Portal > ADX Cluster > Query**
- Populate tables schema in your database by running code in “**opcua-populate-adx-tables.kql**”.
  - Need to run each command that starts with a dot “.” separately

```
.set-or-replace AssetInventory <|  
iiotdata  
| where isnotempty( Asset)  
| distinct Asset  
| extend Capacity=real(1200)
```

```
.set-or-replace ShiftSchedule <|  
range times from datetime(2022-05-01) to datetime(2023-05-01) step 8h  
| serialize start=times, end=next(times)  
| extend dummy=1  
| join kind=inner (AssetInventory | extend dummy=1) on dummy  
| project Asset,ShiftStart=start,ShiftEnd=end
```

# Configure Data Export for OPCDevice – Verify Export

- Run following to see OEE calculation

```
AssetPerformance("", now(-1h), now())
| where ItemCountGood >0
```

Asset	EnqueuedTime	Status	ItemCountGood	ItemCountBad	OEE	Availability	Quality	Performance	ShiftStartMarker
> opc.tcp...	2022-06-02 20:03...	0	207	6			0	0.971830...	
> opc.tcp...	2022-06-02 20:04...	0	1,109	52			0	0.957787...	
> opc.tcp...	2022-06-02 20:05...	0	1,000	54			0	0.953871...	
> opc.tcp...	2022-06-02 20:06...	1	1,236	68		0.00037778617302606723		0.951768...	
> opc.tcp...	2022-06-02 20:07...	1	1,045	66		0.0007552870090634441		0.949205...	
> opc.tcp...	2022-06-02 20:08...	1	1,185	69		0.0011325028312570782		0.948335...	
> opc.tcp...	2022-06-02 20:09...	1	1,160	50		0.0015094339622641509		0.950047...	
> opc.tcp...	2022-06-02 20:10...	1	399	20		0.001886080724254998		0.950168...	
> opc.tcp...	2022-06-02 20:07...	1	522	19		0.0007552870090634441		0.964879...	
> opc.tcp...	2022-06-02 20:08...	1	1,269	65		0.0011325028312570782		0.9552	
> opc.tcp...	2022-06-02 20:09...	1	1,234	78		0.0015094339622641509		0.949168...	
> opc.tcp...	2022-06-02 20:10...	1	409	18		0.001886080724254998		0.950193...	

# (Optional) Configure Data Export for PLCDevice – Create Destination

- Goto **Data Export > Destinations > Add Destination**
  - Destination Name: **ADXPLC**
  - ADX Cluster URL: **https://{adxname}.eastus2.kusto.windows.net**
  - Database Name: **iotcholdb**
  - Table Name: **iiotdataplc**
  - Authorization: **System-assigned Managed Identity**
- Click **Save**

Destinations > ADXPLC

ADXPLC

To create a new destination, select a destination type and enter the connection information. [Learn more](#)

Destination type \*

Azure Data Explorer

Cluster URL \*

https://{redacted}.eastus2.kusto.windows.net

Database name \*

iotcholdb

Table name \*

iiotdataplc

Authorization \*

System-assigned managed identity

Configure the following fields associated with the managed identity.  
[Learn more](#)

# (Optional) Configure Data Export for PLCDevice – Create Export

- Goto **Data Export > Exports > Add Export**
  - Name of export: **ADXPLCExport**
  - Type of data to export: **Telemetry**
  - Destination: **ADXPLC**
- Click **+Filter**
- Select
  - Name=**Device Template Name**
  - Operator=**Equals**
  - Value=**PLCDevice**
- Click **Transform**
- Paste contents of “**opcplc-tx-query.txt**” into “**2. Build transformation query**”
- Click **Save**
- Wait for **Export Status** to be “**Healthy**”

Exports > ADXPLCExport

ADXPLCExport

☒ Enabled

**Data**

All of your devices will export data unless you add filters to narrow things down. [Learn more](#)

Type of data to export

Telemetry

Export the data if all of the conditions are true

Name \* Device template name Operator \* Equals

Value \* PLCDevice

[+ Filter](#) [+ Message property filter](#)

**Enrichments**

Add additional information to your export. This will appear as a key value pair in exported messages. [Learn more](#)

[+ Custom string](#) [+ Property](#)

**Destinations**

Select destinations for your export. If you can't find your destination, [create a new one](#).

Destination *	Data transformation	Export status	Details
<span>ADXPLC</span>	<a href="#">Edit</a>	<span>✓ Healthy</span>	

[+ Destination](#)

## (Optional) Configure Data Export for PLCDevice – Verify Export

- Goto **Azure Portal > ADX Cluster > Query**
- Run following to ensure data coming into ADX

```
iiotdataplc
```

	EnqueuedTime	GatewayId	Asset	Status	AlternatingBoolean	RandomSignedInt32	FastUInt1	NegativeTrendData	PositiveTrendData	SlowUInt1	SpikeData	applicationId	com
>	2022-06-03 16:52...	plc1	null							8,573		26cb1c6c-809d...	null
>	2022-06-03 16:52...	plc1	null		false	1,467,356,015	85,745	-1,614.7	1,814.9		-77.0513242...	26cb1c6c-809d...	null
>	2022-06-03 16:52...	plc1	null							8,574		26cb1c6c-809d...	null
>	2022-06-03 16:52...	plc1	null							8,575		26cb1c6c-809d...	null
>	2022-06-03 16:52...	plc1	null							8,576		26cb1c6c-809d...	null
>	2022-06-03 16:52...	plc1	null							8,577		26cb1c6c-809d...	null
>	2022-06-03 16:52...	plc1	null							8,578		26cb1c6c-809d...	null

# Power BI Report - Configure

- Open **OEEReport.pbix** file with Power BI Desktop
- Goto **Home > Transform Data > Edit Parameters**
- Optionally set date range (**FromDate/ToDate**)
- Set “**ADXURL**” to the URL of your ADX instance
- Set “**ADXDB**” to the database name in your ADX instance (iotcholdb)
- Click **Save**
- Click **Apply Changes**

## Edit Parameters

Asset

FromDate

ToDate

ADXURL

ADXDB

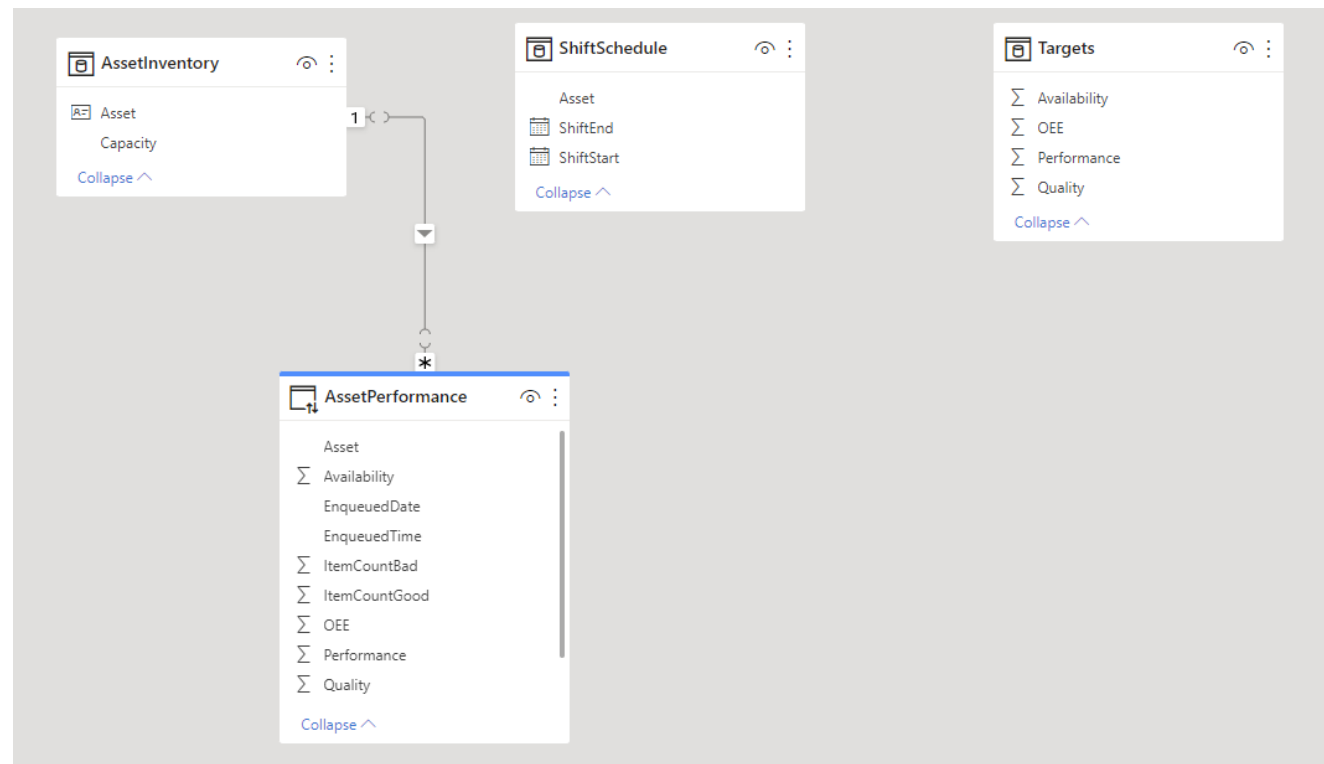
OK

Cancel

# Power BI Report – Data Model

Power BI Report contains 4 tables

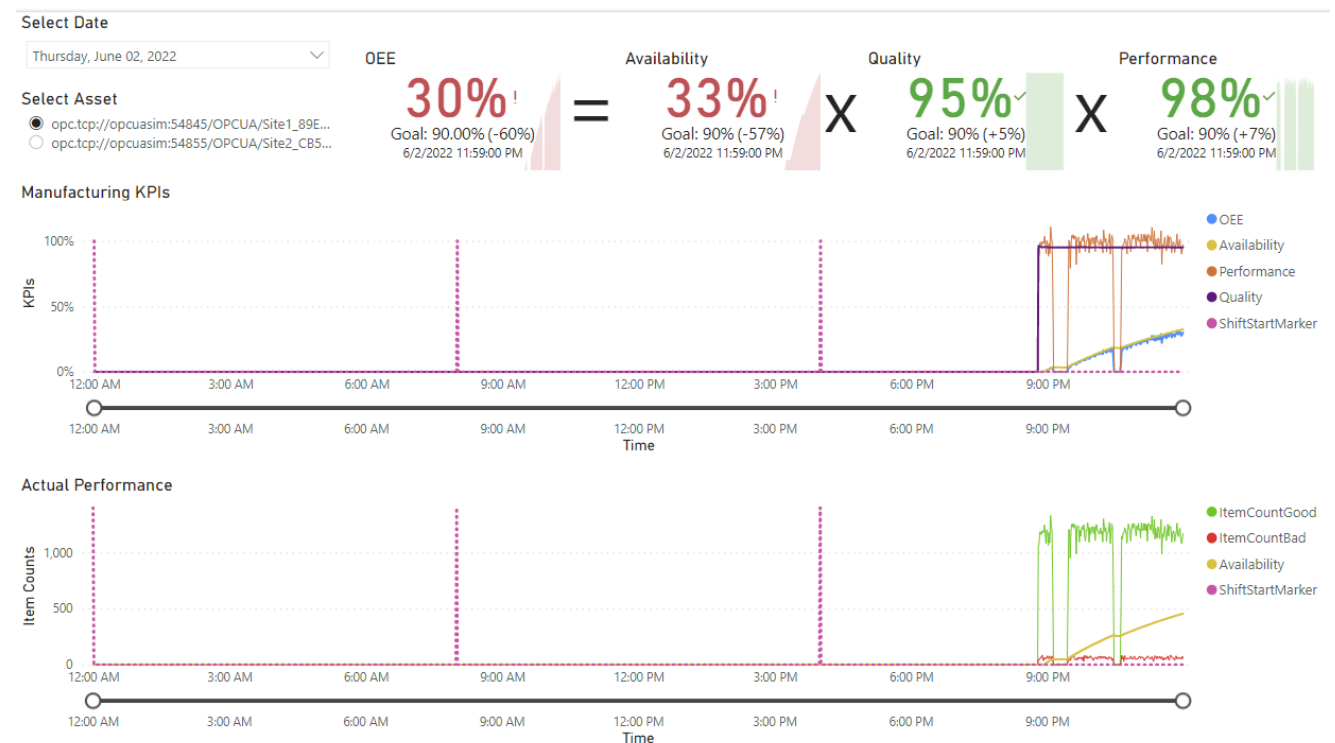
- **AssetPerformance**: DirectQuery mode. Populated running ADX function AssetPerformance.
- **AssetInventory**: Import mode. Asset master data. Contains Asset Name and Capacity (items/min)
- **ShiftSchedule**: Import mode. Contains shift information for each asset. Each day has 3 shifts of 8h.
- **Targets**: Constant table. Contains OEE KPIs' targets. 90% for each KPI.





# Power BI Report – How it works

- Report shows OEE for
  - The selected date from top left. Only a single day is selectable
  - The selected asset from top left. Only one asset at a time.
- A day consist of 3 shifts of 8h each
- OEE resets/starts a new calculation at every shift
- Shift boundaries are marked as dashed vertical purple lines in bottom two graphs
- Top line KPIs show only the last shift of the day



# Power BI Report - Publish

- Select **Publish**
- Select workspace (**My Workspace**)
- Click **OK**
- Click “**Open ‘OEEReport.pbix’ in Power BI**”

Publishing to Power BI

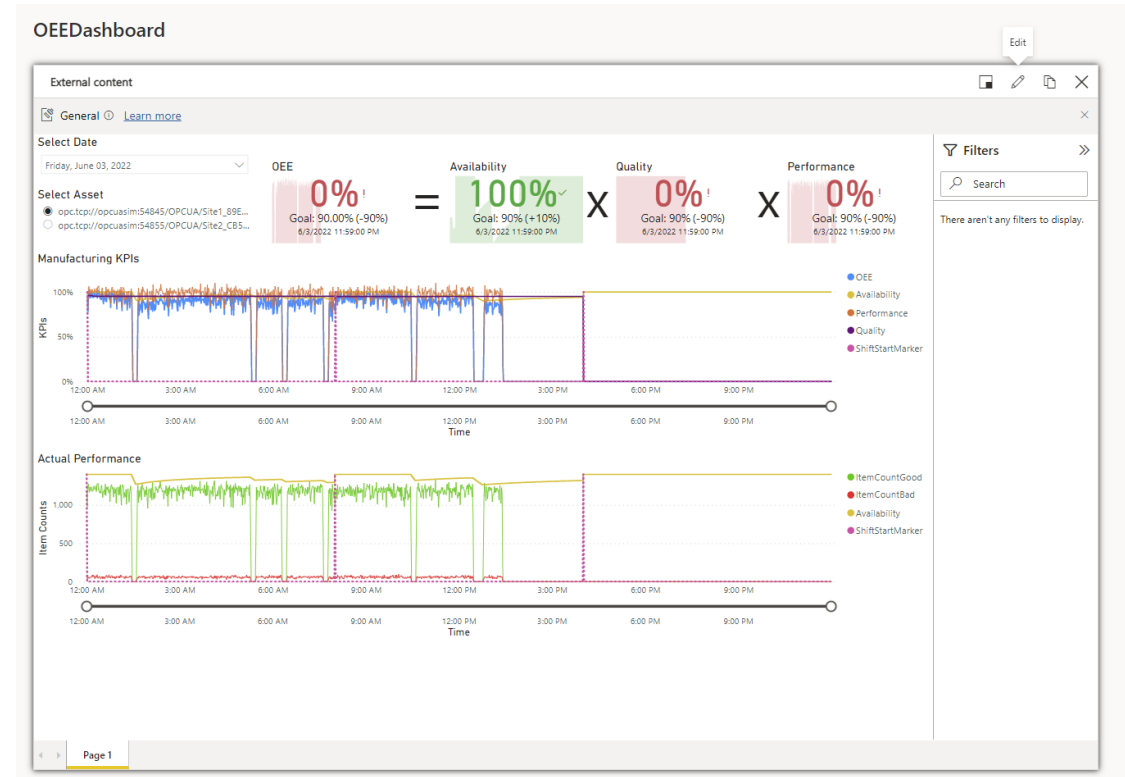
✓ Success!

[Open 'OEEReport.pbix' in Power BI](#)

- Goto **File > Embed Report > Website or Portal**
- Copy the first link

# Power BI Report – Add Power BI Tile

- Go back to IOTC App
- Click **Dashboards > Dashboard Catalog > New Dashboard**
- Name the dashboard and **Save**
- **Edit Dashboard**
- Add a new “**External Content**” tile
- Click **Edit**
- Set “**Source**” to the link you copied at above



# Next Steps

