e!COCKPIT Application Note

Building

Automation

WAGO-I/O-PRO V2.3

WAGO-I/O-SYSTEM 750

WagoAppCloud

Version 1.4.5

WAGO Kontakttechnik GmbH & Co. KG

Hansastraße 27
D-32423 Minden

Tel.:	+49 (0) 571/8 87 – 0
Fax:	+49 (0) 571/8 87 – 1 69
E-Mail:	info@wago.com
Web:	http://www.wago.com

Technical Support

Tel.:	+49 (0) 571/8 87 – 4 45 55
Fax:	+49 (0) 571/8 87 – 84 45 55
E-Mail:	support@wago.com

Notes about this Documentation

Copyright

Number Notation

Table 1: Number Notation

Number code	Example	Note
Decimal	100	Normal notation
Hexadecimal	0x64	C notation
Binary	'100' '0110.0100'	In quotation marks, nibble separated with dots (.)

Font Conventions

Table 2: Font Conventions

Font type	Indicates
italic	Names of paths and data files are marked in italic-type. e.g.: *C:\Programme\WAGO-I/O-CHECK*
Menu	Menu items are marked in bold letters. e.g.: **Save**
>	A greater-than sign between two names means the selection of a menu item from a menu. e.g.: **File** > **New**
Input	Designation of input or optional fields are marked in bold letters, e.g.: **Start of measurement range**
“Value”	Input or selective values are marked in inverted commas. e.g.: Enter the value “4 mA” under **Start of measurement range**.
[Button]	Pushbuttons in dialog boxes are marked with bold letters in square brackets. e.g.: **[Input]**
[Key]	Keys are marked with bold letters in square brackets. e.g.: **[F5]**

Symbols

⚠ DANGER

Personal Injury!
Indicates a high-risk, imminently hazardous situation which, if not avoided, will result in death or serious injury.

⚠ DANGER

Personal Injury Caused by Electric Current!
Indicates a high-risk, imminently hazardous situation which, if not avoided, will result in death or serious injury.

⚠ WARNING

Personal Injury!
Indicates a moderate-risk, potentially hazardous situation which, if not avoided, could result in death or serious injury.

⚠ CAUTION

Personal Injury!
Indicates a low-risk, potentially hazardous situation which, if not avoided, may result in minor or moderate injury.

NOTICE

Damage to Property!
Indicates a potentially hazardous situation which, if not avoided, may result in damage to property.

NOTICE

Damage to Property Caused by Electrostatic Discharge (ESD)!
Indicates a potentially hazardous situation which, if not avoided, may result in damage to property.

Note

Important Note!
Indicates a potential malfunction which, if not avoided, however, will not result in damage to property.

Information

Additional Information:
Refers to additional information which is not an integral part of this documentation (e.g., the Internet).

Legal Bases

Subject to Change

WAGO Kontakttechnik GmbH & Co. KG reserves the right to make any alterations or modifications that serve to increase the efficiency of technical progress. WAGO Kontakttechnik GmbH & Co. KG owns all rights arising from granting patents or from the legal protection of utility patents. Third-party products are always mentioned without any reference to patent rights. Thus, the existence of such rights cannot be excluded.

Personnel Qualification

The use of the product described in this document is exclusively geared to specialists having qualifications in PLC programming, electrical specialists or persons instructed by electrical specialists who are also familiar with the appropriate current standards.

Moreover, the persons cited here must also be familiar with all of the products cited in this document, along with the operating instructions. They must also be capable of correctly predicting any hazards which may not arise until the products are combined.

WAGO Kontakttechnik GmbH & Co. KG assumes no liability resulting from improper action and damage to WAGO products and third-party products due to non-observance of the information contained in this document.

Limitation of Liability

This documentation describes the use of various hardware and software components in specific example applications. The components may represent products or parts of products from different manufacturers. The respective operating instructions from the manufacturers apply exclusively with regard to intended and safe use of the products. The manufacturers of the respective products are solely responsible for the contents of these instructions.

The sample applications described in this documentation represent concepts, that is, technically feasible application. Whether these concepts can actually be implemented depends on various boundary conditions. For example, different versions of the hardware or software components can require different handling than that described here. Therefore, the descriptions contained in this documentation do not form the basis for assertion of a certain product characteristic.

Responsibility for safe use of a specific software or hardware configuration lies with the party that produces or operates the configuration. This also applies when one of the concepts described in this document was used for implementation of the configuration.

WAGO Kontakttechnik GmbH & Co. KG is not liable for any actual implementation of the concepts.

Table of Contents

1 Description

In this day and age, it is always more important to make data from an industrial controller available in a cloud. In this application note, the steps are explained how the PFC100/PFC200 – hereinafter referred to as a PFC only – can be enabled to send data from the PLC program to a cloud service and to receive commands from the cloud depending on the cloud service being used.

1.1 Introduction

The Industrial Internet of Things and Services (IoT) aims smarter products and applications by collecting and understanding data provided by the "things" using internet and Cloud technology.

These things are typically the sensors, actors and other microprocessor equipped devices used for automation. However, these devices typically don't have internet interfaces. Thus, gateways are required to connect those devices to the Cloud via the internet.

The WAGO Programmable Field Controller (PFC) is the perfect base for an Industrial Internet of Things gateway. It can access data from most field protocols due the large variety of modules. It can access the internet due to the built-in Ethernet and mobile interfaces (model depended).

Microsoft Azure and Amazon Web Services are Cloud platforms which provides services for the implementation of IoT applications. It provides e.g. services to retrieve thousands of telemetry messages per second, data archiving and analysis as well as web and mobile user interfaces for visualization.

Cloud Connectivity enables a PLC program, which is running on the PFC, to communicate with such a Cloud platform.

1.2 Overview

Following diagram gives an overview about the Cloud Connectivity

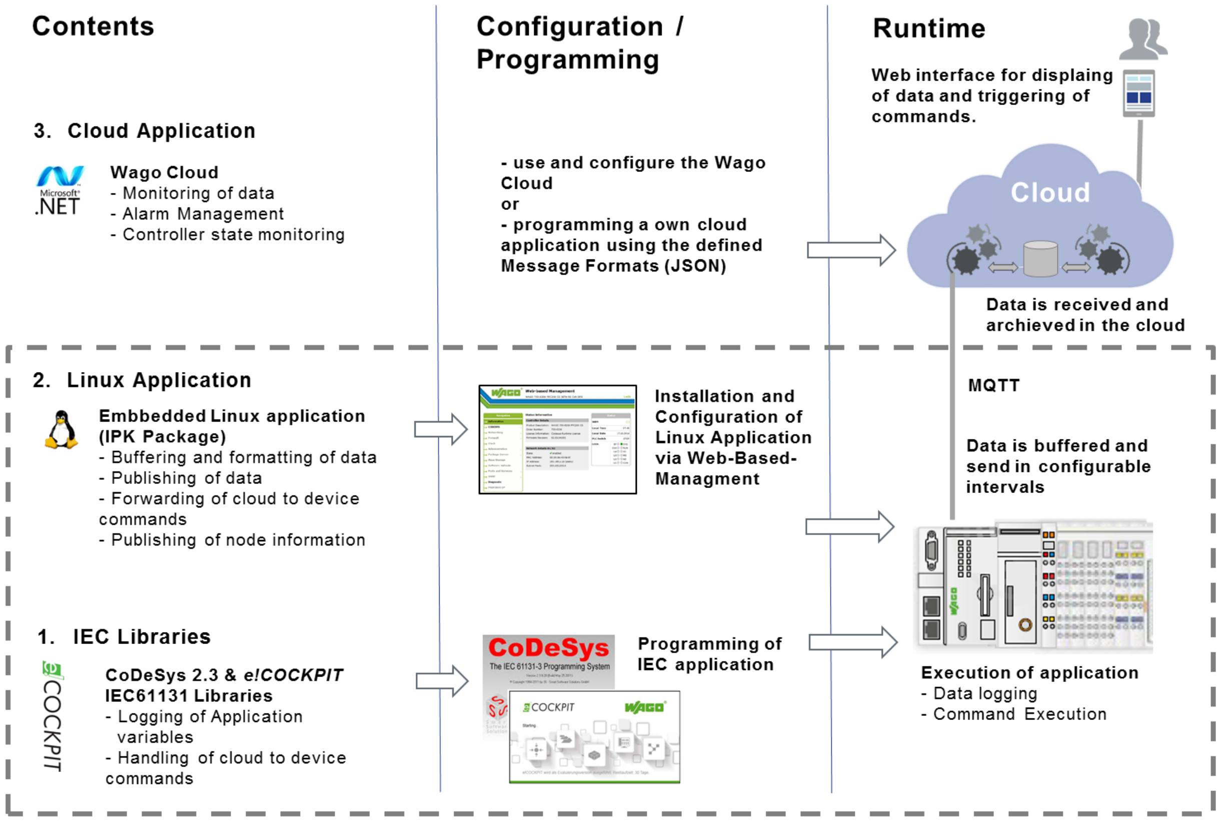

Contents
Configuration / Programming
Runtime
Web interface for displaying of data and triggering of commands.
3. Cloud Application
Wago Cloud
- Monitoring of data
- Alarm Management
- Controller state monitoring
Microsoft .NET
- use and configure the Wago Cloud
or
- programming a own cloud application using the defined Message Formats (JSON)
Cloud
Data is received and archieved in the cloud
2. Linux Application
Embbedded Linux application (IPK Package)
- Buffering and formatting of data
- Publishing of data
- Forwarding of cloud to device commands
- Publishing of node information
Installation and Configuration of Linux Application via Web-Based-Managment
MQTT
Data is buffered and send in configurable intervals
1. IEC Libraries
COCKPIT
CoDeSys 2.3 & e!COCKPIT IEC61131 Libraries
- Logging of Application variables
- Handling of cloud to device commands
CoDeSys
The IEC 61131-3 Programming System
Programming of IEC application
Execution of application
- Data logging
- Command Execution

Figure 1-1: Overview about Cloud Connectivity

The Cloud Connectivity consists of IEC libraries (for e!Cockpit and Codesys 2.3) and related Linux application, which is running on PFC as a daemon. The main features of both are described in the following chapters.

The cloud application will not be described in this document. Most of the customers would implement the cloud application by themselfes according to their special needs and with the help of the documented message formats in the specification of the WAGO Protocol. There is an already an existing cloud application named WAGO Cloud which can be used by the customers of WAGO. WAGO Cloud is based on the Microsoft Azure Platform. Chapter 4 has a description about connecting PFC to a cloud service.

1.2.1 Operation mode

Cloud Connectivity has following operation modes:

- **WAGO Protocol**
  Enables a PLC program to capture and send its process data comfortably to the cloud application. Furthermore the PLC can define and publish its custom commands. Later on the cloud application can trigger the executions of those commands. WAGO Protocol can be used for any cloud platform: currently it is already being used for WAGO Cloud.

  The operation mode WAGO Protocol can be used with all selectable cloud platforms.

- **Native MQTT**
  Here the PLC program turns to a “thing” (IoT) or rather MQTT-Client. It will retain the full control and responsibility for every outgoing message and its format.

  The operation mode Native MQTT is available for the following cloud platforms:
  - Amazon Web Services (AWS)
  - SAP IoT Services
  - IBM Cloud
  - MQTT AnyCloud

- **Sparkplug**
  Sparkplug (Sparkplug payload B) is a specification of Cirrus Link in which the communication of the MQTT messages is defined in more detail. The exact designation of the MQTT topics, the structure and content of the MQTT payload and the sequence of message packages are defined in the specification. By supporting the Sparkplug specification, data can be sent from the PFC to SCADA systems such as Ignition. Process data can be recorded and configured in the PLC program, similar to the operation mode WAGO protocol. Furthermore, commands can also be defined in the PLC program and sent from the SCADA system to the PFC.

  The Linux application ensures that the MQTT messages support the Sparkplug specification.

  The operation mode Sparkplug is available for the following cloud platforms:
  - Amazon Web Services (AWS)
  - MQTT AnyCloud

It is not possible to use multiple operation modes simultaneously at runtime: that means the PLC developer has to make a decision and make further implementation based on it.

Following diagram shows exemplary the usage of the operation modes.

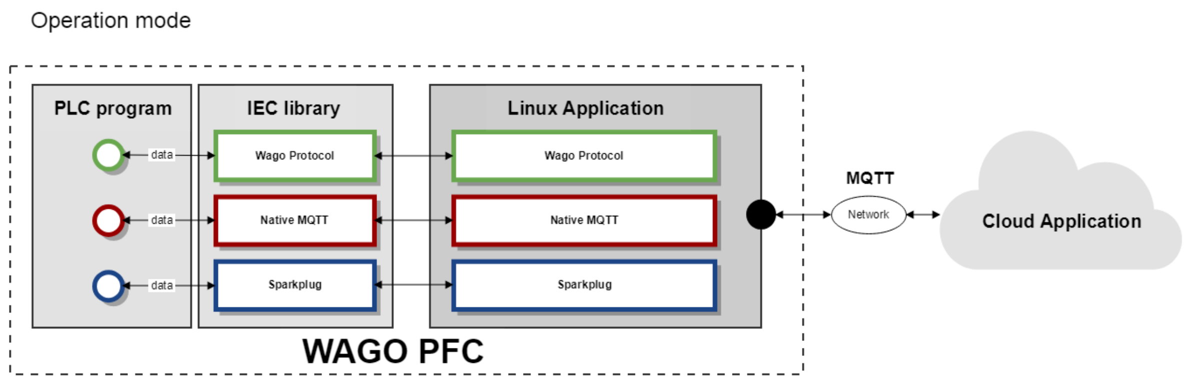


Figure 1-2: Overview operation modes

1.2.2 IEC Libraries

WagoAppCloud (for e!Cockpit) and WagoLibCloud (for Codesys2.3) are IEC 61131 libraries to be used when implementing PLC programs for IoT scenarios. Within the libraries there are different function blocks to be used for different operation mode.

Operation mode WAGO Protocol

Following example programs show the usage of IEC libraries:

- *WagoAppCloud_FbCollectionLogger_Example1.ecp, WagoLibCloud_FbCollectionLogger_Example1.pro*
  Variables of a PLC program are grouped into different collections. The values of variables are sampled on a time cycle and forwarded to the cloud.

- *WagoAppCloud_FbCommandListener_Example1.ecp, WagoLibCloud_FbCommandListener_Example1.pro*
  Variables of a PLC program are grouped into different collections. The values of variables are sampled and sent cyclicly to the cloud. PLC program also implements a custom command and shows how it can be handled.

- *WagoAppCloud_FbCollectionLogger_Example2.ecp, WagoLibCloud_FbCollectionLogger_Example2.pro*
  Variables of a PLC program are grouped into different collections. Some values of variables are sampled by certain event and sent to the cloud.

The operation mode WAGO Protocol used the following function blocks

- FbCollectionLogger
- FbCommandConfigurator
- FbCommandListener
- FbCommandResponder
- FbStatus_WagoProtocol

Operation mode Native MQTT

Note

Important note!
The operation mode Native MQTT is available for the PLC library WagoAppCloud (e!Cockpit). WagLibCloud doesn't support this operation mode.

Following example programs show the usage of IEC libraries:

- *WagoAppCloud_FbPublishMQTT_Example1.ecp*
  PLC program creates a primitive JSON document und publishes it via MQTT
- *WagoAppCloud_FbSubscribeMQTT_Example1.ecp*
  PLC program subscribe data via MQTT from the cloud

The operation mode Native MQTT used the following function blocks

- FbPublishMQTT
- FbSubscribeMQTT
- FbStatus_NativeMQTT

Operation mode Sparkplug

The configuration of the data points in the IEC application in the operation mode Sparkplug is very similar to that of the operation mode WAGO Protocol.

Response form a command (FbCommandResponder) is not supported by the Sparkplug specification by the operation mode Sparkplug.

Following example programs show the usage of IEC libraries:

- *WagoAppCloud_FbCommandListener_Sparkplug.ecp*
  Variables of a PLC program are grouped into different collections. Some values of variables are sampled by certain event and sent to the cloud. PLC program also implements a custom command and shows how it can be handled for the operation mode Sparkplug.

The operation mode Sparkplug used the following function blocks

- FbCollectionLogger
- FbCommandConfigurator
- FbCommandListener
- FbStatus_WagoProtocol

1.2.3 Linux Application

The supported operation modes of Cloud Connectivity originate from the Linux Application.

The Linux Application is a deamon running on the PFC. It is responsible for data flow between PLC program and cloud application. It is connected to the IEC Library using Inter-process communication (IPC) and also communicates with the Cloud. Data transmission to the Cloud is done by using the MQTT protocol and is encrypted using Transport Layer Security (TLS).

The Linux Application can be configured to connect different Cloud platforms which support the MQTT protocol. The configuration is quite easy step by using the Web Based Management (WBM). Operation mode is configurable there as well.

The Linux Application caches data coming from the PLC program to avoid data loss in case network connection to the Cloud got interrupted. Linux Application takes care about automatic reconnect.

Operation mode WAGO Protocol

Linux Application can provide information about the PFC and its state:

1. **Device Information**
   Linux Application publishes PFC specific information to the Cloud when connecting.
2. **Device Status**
   Linux Application publishes PFC specific status information on state changes to the Cloud.

For the operation mode WAGO Protocol caching can be configured within Web Based Management (WBM) either in RAM or on a storage medium (SD Card).

Operation mode Native MQTT

PLC program will use the Linux Application as a Gateway to the MQTT-Broker.

Operation mode Sparkplug

The linux application forwards the data from the PLC program to the broker in a compressed data format ‘Goggle Protocol Buffer’. The Linux application ensures compliance with the Sparkplug specification.

Note

Important note!
The operation mode Sparkplug is available on 0750-821x devices.

Please check that the Sparkplug license is added to the device. 30 days Sparkplug could be test for free with the test license.

2 Used material

2.1 Libraries

Library	Description
e!*COCKPIT*: WagoAppCloud_x_x_x_x	The library provides Cloud Connectivity on the PLC side for ***e!****COCKPIT*.
CODESYS 2.3: WagoLibCloud	The library provides Cloud Connectivity on the PLC side for CODESYS 2.3.

2.2 Devices

Provider	Quantity	Description	Order number
WAGO Kontakttechnik GmbH & Co. KG	1	PFC100 / PFC200 Touch Panel 600	0750-8xxx / xxxx-xxxx 0762-4x0x / xxxx-xxxx 0762-5x0x / xxxx-xxxx 0762-6x0x / xxxx-xxxx

2.3 Firmware and PLC library versions

Note

Important note!
Please check if the appropriate library for the used firmware is installed on your PFC. If the PLC library does not match the Cloud Connectivity version, the Cloud Connectivity will be terminated.

Firmware	Cloud Connectivity	e!Cockpit Bibliotheken	CODESYS 2.3 Bibliothek
FW11 v020825	1.0.0	WagoAppCloud v1.2.0.7	WagoLibCloud v1.3.2 WagoLibCloud_internal
FW11 Patch1 v020831	1.0.0	WagoAppCloud v1.2.0.7	WagoLibCloud v1.3.2 WagoLibCloud_internal
FW11 Patch2 v020835	1.1.0	WagoAppCloud v1.2.0.7	WagoLibCloud v1.3.2 WagoLibCloud_internal
FW12 v030035	1.2.0	WagoAppCloud v1.3.0.7	WagoLibCloud_02 v2.2.2 WagoLibCloud_internal_02
FW12 Patch1 v030039	1.2.2	WagoAppCloud v1.3.0.7	WagoLibCloud_02 v2.2.2 WagoLibCloud_internal_02
FW13 Patch1 v030107	1.2.2	WagoAppCloud v1.3.0.8	WagoLibCloud_02 v2.2.2 WagoLibCloud_internal_02

3 IEC Application

Note

Installation of Sample Projects for *e!COCKPIT*

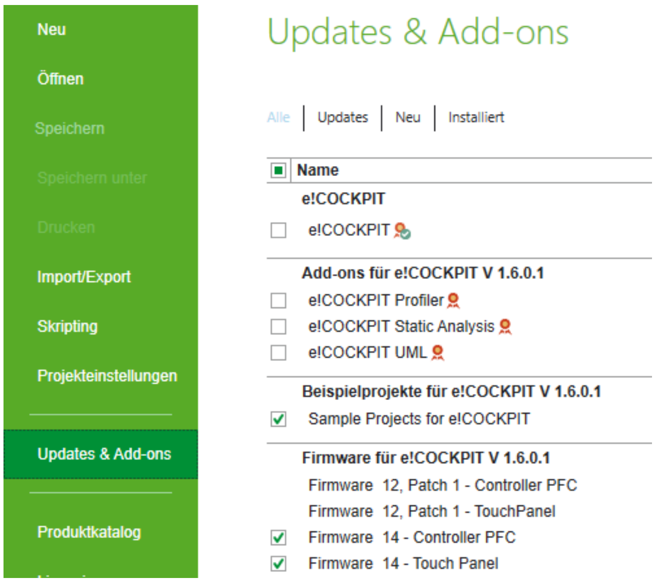

Neu
Öffnen
Speichern
Speichern unter
Drucken
Import/Export
Skripting
Projekteinstellungen
Updates & Add-ons
Produktkatalog
Updates & Add-ons
Alle
Updates
Neu
Installiert
Name
e!COCKPIT
e!COCKPIT
Add-ons für e!COCKPIT V 1.6.0.1
e!COCKPIT Profiler
e!COCKPIT Static Analysis
e!COCKPIT UML
Beispielprojekte für e!COCKPIT V 1.6.0.1
Sample Projects for e!COCKPIT
Firmware für e!COCKPIT V 1.6.0.1
Firmware 12, Patch 1 - Controller PFC
Firmware 12, Patch 1 - TouchPanel
Firmware 14 - Controller PFC
Firmware 14 - Touch Panel

Figure 3-1: ***e!****COCKPIT* Sample Projects

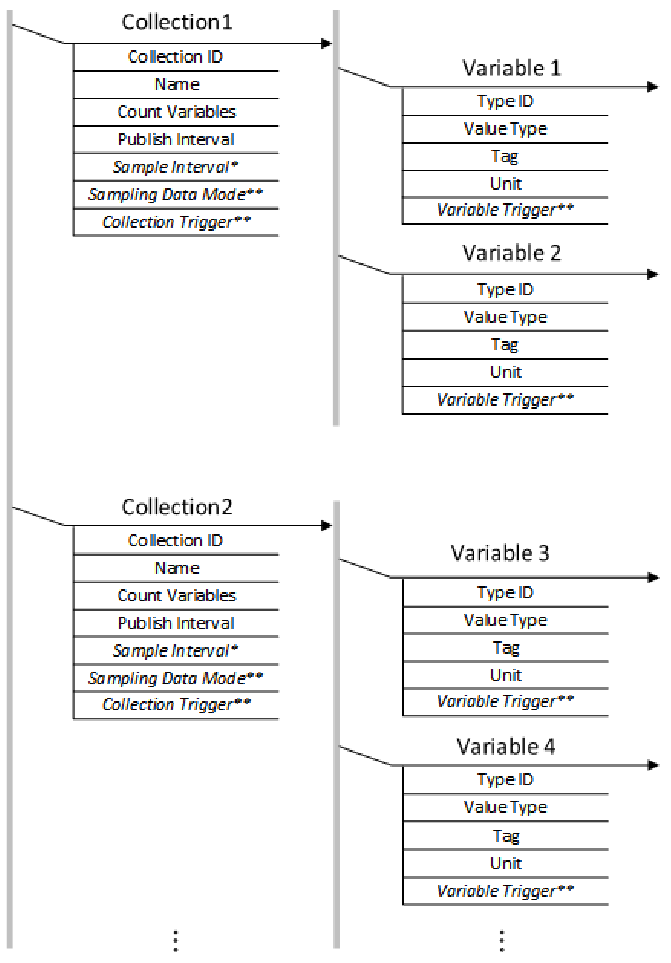
Sample programs can be called up from the ***e!****COCKPIT* Backstage view by clicking the **Updates & Add-ons** button in the navigation bar.

This chapter describes in details how a PLC program for the operation mode „WAGO Protocol“ can be implemented using e!Cockpit. It is helpful to check the referenced example programs as well when implementing own PLC program from scratch. In the following, the creation of a PLC application with the WAGO Protocol version 1.2 is described (further information on the WAGO Protocol and the versions is described in the WAGO Protocol specification).

3.1 Data transmission to cloud

Telemtry is done using the Function Block *FbCollectionLogger*. This function block provides implementation of two different ways to transmit data to the cloud. The one way is to send data cyclicly by using a constant time interval, the other is by doing it on some event. Depending from the kind of required transmission the related library parameters must be configured differently. The configuration of the related parameters will also be sent to the cloud, so that cloud application is aware of transmitted data, which mostly will be time series.

The related function block takes an array of type *typCollection* and the number of collections as arguments. A Collection is a container for a set of variables which are sampled and transmitted as a group. It is identified by a unique Id and requires a name (for display in the Cloud).



* parameter is only for cyclical data transmission necassary
** parameter is only for eventbased data transmission necassary

Figure 3-2: Logged variables as a group in a collection

Note

Important note!
Please consider the name conventions for Tags and Collection within the PLC program when using the WAGO Cloud. The PLC program developer is responsible for observance for the naming conventions (see chapter 4.2.1.1). If the naming conventions are disregarded, there will be problems.

With collections the PLC programmer can structure the variables clearly. This is important for the cyclical data transmission. So the PLC programmer can set different sample and publish intervals for each collection. Therefore it is possible to reduce the amount of data to be transmitted. The programmer has to decide which variable are needed in the cloud in which interval.

For each Collection, an individual sample interval and publish interval can be defined.

The sample interval specifies the time between two samples of variable values. The publish interval specifies the time between the attempts for sending the samples to the Cloud.

Both sample interval and publish interval can be changed from the Cloud.

Note

Important note!
The sample and publish interval must be declared as "RETAIN". Otherwise the updated values are lost after a reboot.

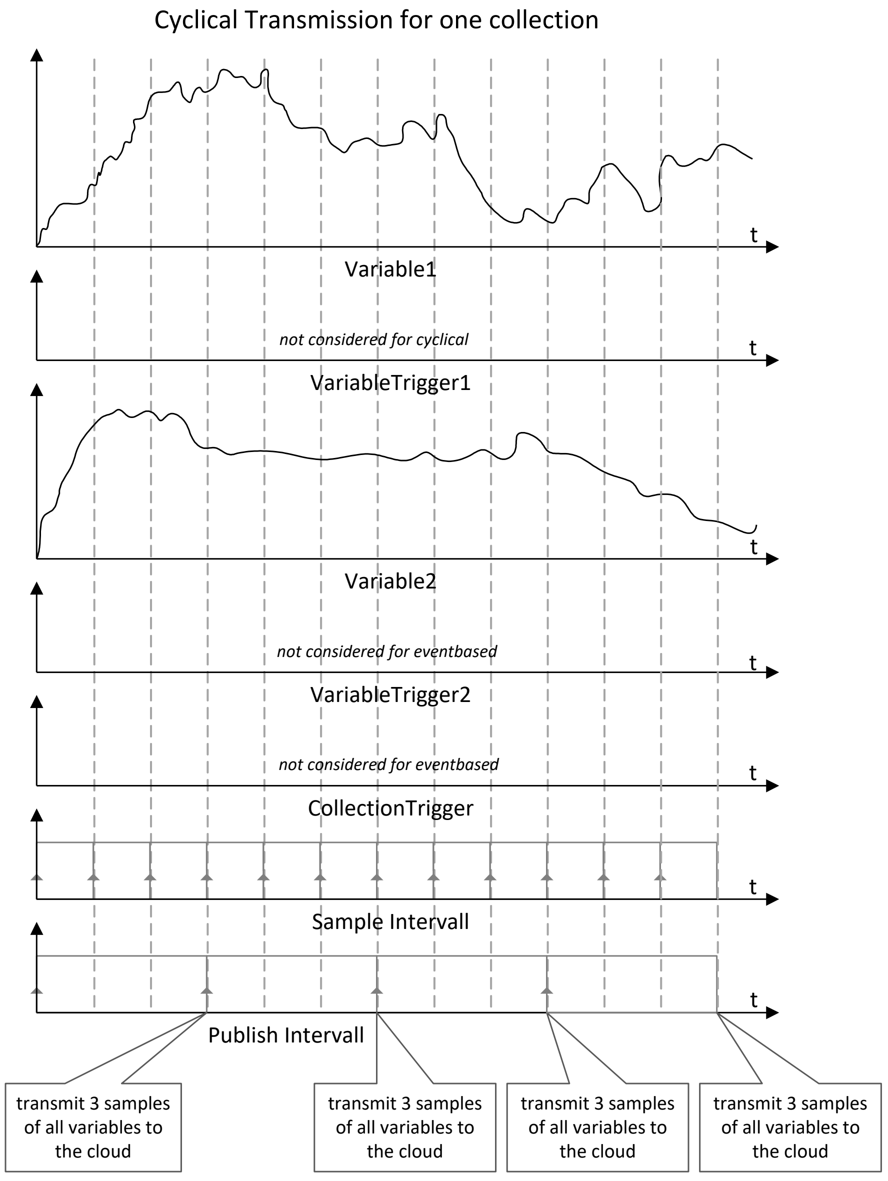

Cyclical Transmission for one collection
Variable1
t
not considered for cyclical
VariableTrigger1
t
Variable2
t
not considered for eventbased
VariableTrigger2
t
not considered for eventbased
CollectionTrigger
t
Sample Intervall
t
Publish Intervall
t
transmit 3 samples of all variables to the cloud
transmit 3 samples of all variables to the cloud
transmit 3 samples of all variables to the cloud
transmit 3 samples of all variables to the cloud

Figure 3-3: Example of cyclical transmission for one collection

In case of eventbased tansmission the parameter *xSamplingDataMode* of the collection must initialized accordingly. The individual options could be initialized independent from the other collections. So one collection could send the data eventbased and another one can send the data cyclicly.

The PLC programmer can trigger the event per collection or per variable. With the *xCollectionTrigger* all variables of the collections will be sampled and forwaded to the cloud. By *xVariableTrigger* only the value of the related variable will be sampled and sent to the cloud. By setting the trigger properly the PLC programmer is responsible for supervising the amount of data to be transferred. For every event the PLC programmer can also define specific timestamp which will be included into the resulted data sample.

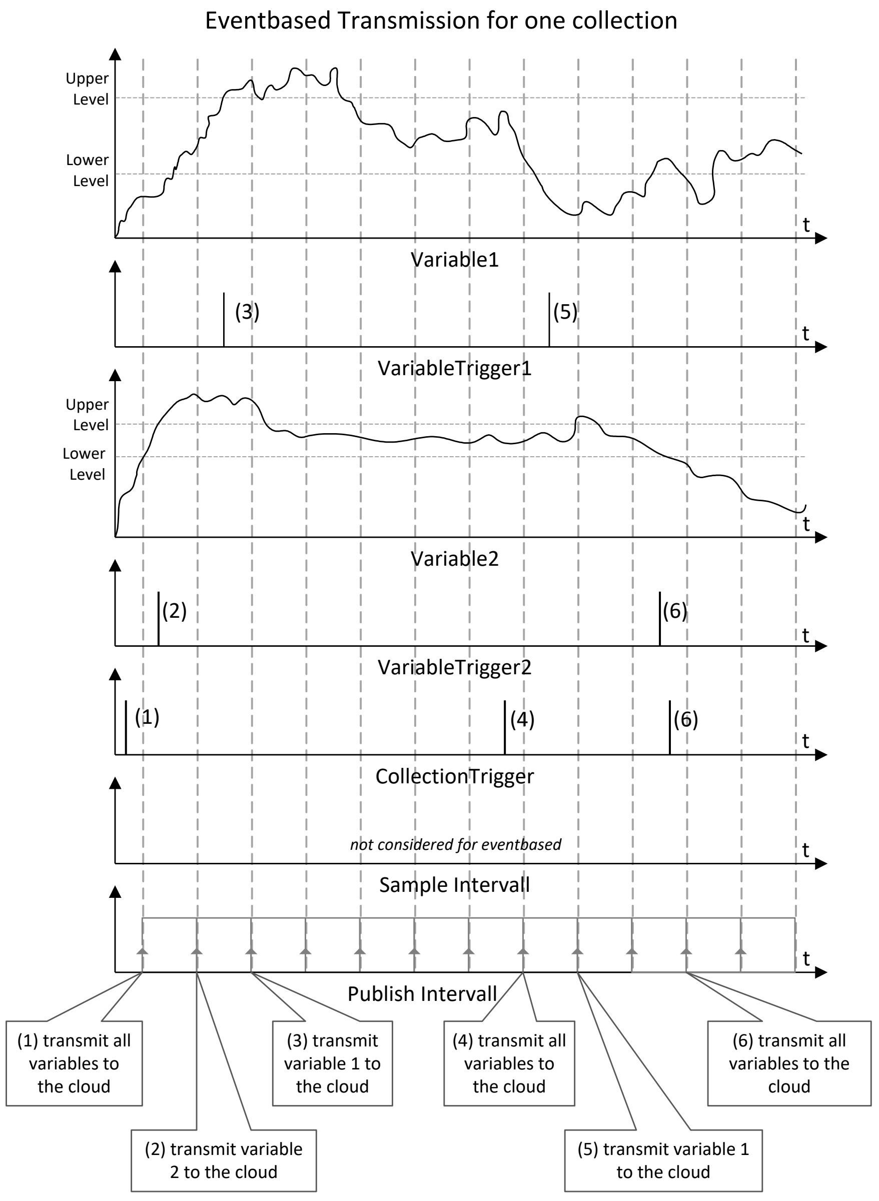

Eventbased Transmission for one collection
Upper Level
Lower Level
t
Variable1
(3)
(5)
VariableTrigger1
Upper Level
Lower Level
Variable2
(2)
(6)
VariableTrigger2
(1)
(4)
(6)
CollectionTrigger
not considered for eventbased
Sample Intervall
Publish Intervall
(1) transmit all variables to the cloud
(3) transmit variable 1 to the cloud
(4) transmit all variables to the cloud
(6) transmit all variables to the cloud
(2) transmit variable 2 to the cloud
(5) transmit variable 1 to the cloud

Figure 3-4: Example of eventbased transmission for one collection[1]

[1] In the example for the eventbased transmission are included limitations for the variables in the PLC application. The event will be set if the value of the variable pass the upper limit or fall below the lower limit. The collection trigger is set from the PLC application independent of the variable. This is an example and shall demonstrate how eventbased data transmission could be used. The PLC programmer can decide in which case it is useful to send the variables to the cloud by setting the trigger.

3.1.1 Limitations

Cyclical data transmission

Note

Important note!
- min. Sample Interval is 100 Milliseconds for Cache mode = RAM

- min. Sample Interval is 1 Second for Cache mode = SD-Card

- min. Publish Interval is 1 Second

Depending on the combination of Sample Interval and Publish Interval the amout of data points, which continuously can be sent to cloud service, is limited. Following limitations have been determined for 2 different scenarios.

	Sample Interval	Publish Interval	Data points
Scenario A	1 Second	1 Minute	10 collections each 120 variables
Scenario B	100 Milliseconds	1 Second	10 collections each 20 variables

The listed limits were measured within the following environment:
- Device: PFC 750-8202
- Network connection: Ethernet 100 Mbit/s
- PLC runtime: e!RUNTIME
- PLC program: minimalistic, 20 Milliseconds cycle time, Tag names have 80 characters
- Both cases produce data flow around 270 KB/s

Eventbased data transmission
The eventbased data transmission wants to give the PLC programmer many opportunities to set an event. Therefore the librarie have less limitations of the data transfer. Finally the PLC programmer is responsible that the PFC isn't overloaded. On average the produce data flow should also don't cross over 270 KB/s.

Note

Important note!
The PublishInterval can not be set shorter than 100 milliseconds.

By using the eventbased data transmission the PLC programmer have to be careful to trigger an event. Otherwise a lot of volume of data could be transfer. Combination with Cache ode = SD-Card is not recommended due to write speed.

Analysis
The parameter "Cache fill level", shown in WBM, can be used to monitor and verify whether your specific productive use case will exceed the limitations described above. If the related value is not continuously increasing for a reasonable period of time then your productive use case should not result in problems. Please also consider some special conditions (e.g. network connection to cloud service is not stable or interrupted) when you make decision about the amount of data points.

3.2 Command Handling

The handling of Cloud-To-Device commands is done using the following Function Blocks

- FbCommandConfigurator
- FbCommandListener
- FbCommandResponder

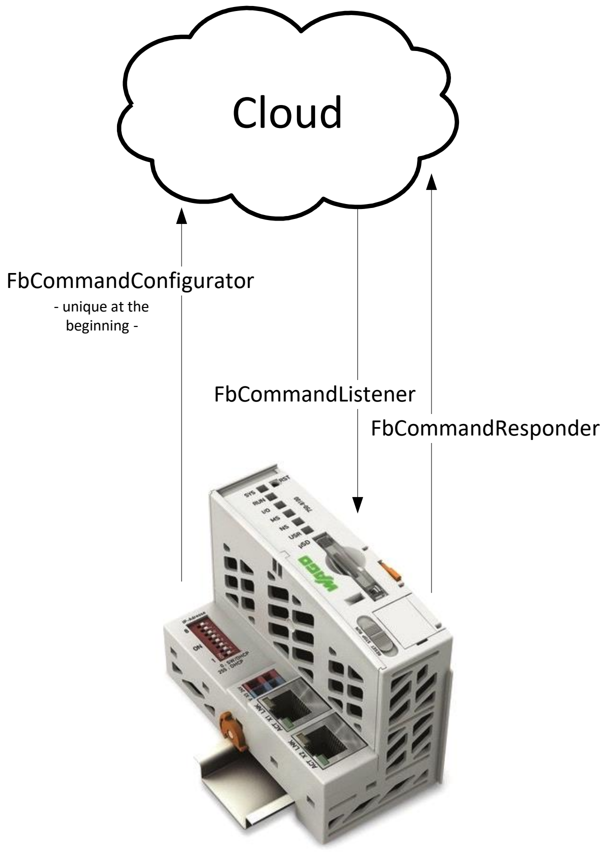


Figure 3-5: Procedure Command Handling

The handling of Cloud-To-Device commands is done by following three steps:

1. Registration of Commands using *FbCommandConfigurator*
   - *FbCommandConfigurator* publishes the provided commands to the Cloud per the information provided by *typCommandDescription* input.
   - The Cloud application can provide a user interface to request the execution of the command based on the *typCommandDescription*.
   - A Command can have up to 16 request and 16 response parameters. These parameters are provided by the struct *typCommandParameterDescription*.

2. Listening for incoming CommandRequests using *FbCommandListener*
   - If a command request arrives from the Cloud the *xCommandReceived* flag is set to true for one cycle.
   - *FbCommandListener* requires a pointer to a *typCommandRequest* struct which contains the information about the command request call.
   - The Command is handled based on the information provided by *typCommandRequest*.

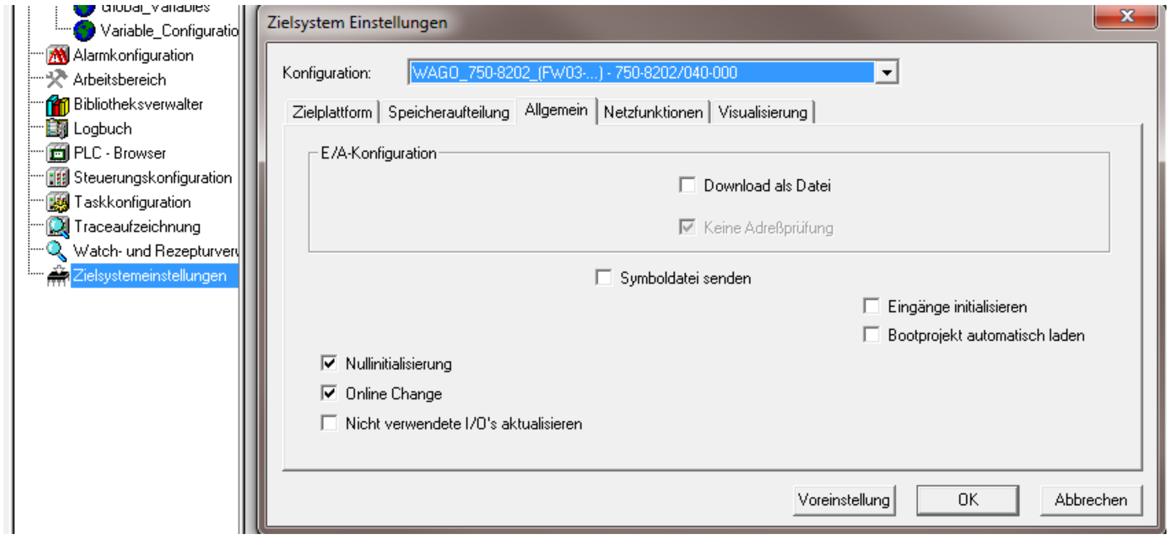
3. Confirmation of command execution using *FbCommandResponder*
   - The execution of every command must be confirmed to the Cloud application by using *FbCommandResponder*.
   - The cloud application shall only send the next command after the previous one was executed and confirmed.
   - The confirmation of command execution should be done by providing a *typCommandResponse*.
   - Response parameters are send to the Cloud by using the same type *typCommandResponse.*

A simple implementation to send commands from the cloud to the device is shown in the example WagoAppCloud_FbCommadListener_Example1.ecp.

3.3 Online Change

Before using the function *online change* by Codesys 2.3 it is necessary to execute the following steps:

1. Activate *online change* in the target settings



Zielsystem Einstellungen

Konfiguration: WAGO_750-8202_(FW03-...) - 750-8202/040-000

Zielplattform | Speicheraufteilung | Allgemein | Netzfunktionen | Visualisierung

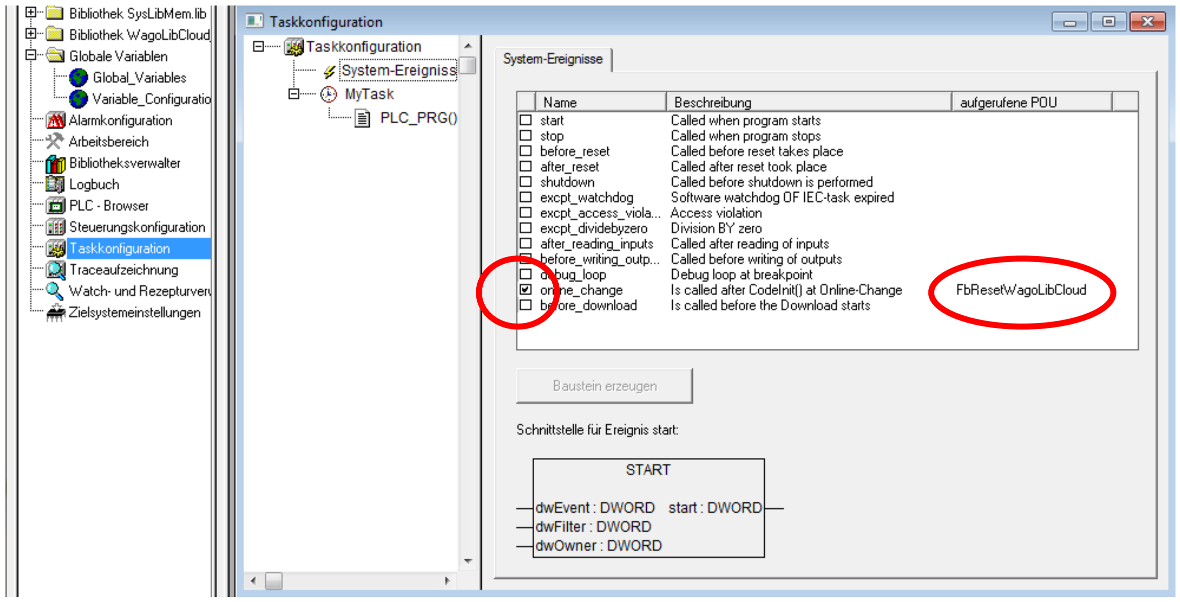
E/A-Konfiguration
- [ ] Download als Datei
- [x] Keine Adreßprüfung

- [ ] Symboldatei senden
- [ ] Eingänge initialisieren
- [ ] Bootprojekt automatisch laden
- [x] Nullinitialisierung
- [x] Online Change
- [ ] Nicht verwendete I/O's aktualisieren

Voreinstellung | OK | Abbrechen

Figure 3-6: Activate online change in the target settings (Codesys 2.3)

2. Activate the system event *online change* in the task configuration and assign the function block FbResetWagoLibCloud from the library WagoLibCloud to this system event



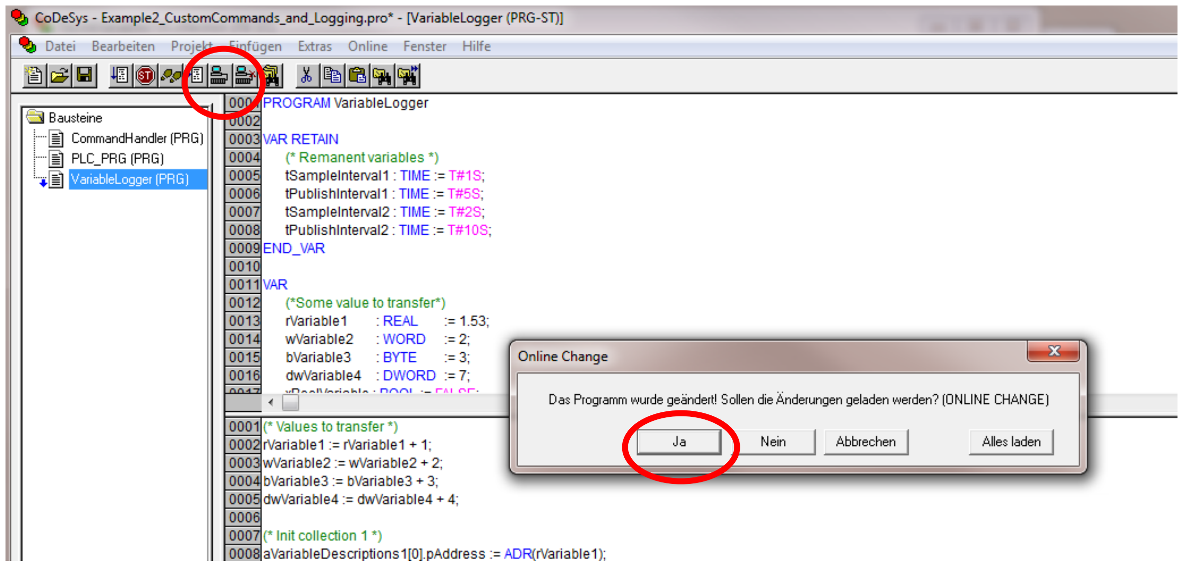
Name	Beschreibung	aufgerufene POU
☐ start	Called when program starts	
☐ stop	Called when program stops	
☐ before_reset	Called before reset takes place	
☐ after_reset	Called after reset took place	
☐ shutdown	Called before shutdown is performed	
☐ excpt_watchdog	Software watchdog OF IEC-task expired	
☐ excpt_access_viola...	Access violation	
☐ excpt_dividebyzero	Division BY zero	
☐ after_reading_inputs	Called after reading of inputs	
☐ before_writing_outp...	Called before writing of outputs	
☐ debug_loop	Debug loop at breakpoint	
☑ online_change	Is called after CodeInit() at Online-Change	FbResetWagoLibCloud
☐ before_download	Is called before the Download starts	

Schnittstelle für Ereignis start:

```
START
dwEvent : DWORD    start : DWORD
dwFilter : DWORD
dwOwner : DWORD
```

Figure 3-7: Activate the system event online change in the task configuration (Codesys 2.3)

3. Compile the program and execute the online change



```
PROGRAM VariableLogger

VAR RETAIN
    (* Remanent variables *)
    tSampleInterval1 : TIME := T#1S;
    tPublishInterval1 : TIME := T#5S;
    tSampleInterval2 : TIME := T#2S;
    tPublishInterval2 : TIME := T#10S;
END_VAR

VAR
    (*Some value to transfer*)
    rVariable1    : REAL    := 1.53;
    wVariable2    : WORD    := 2;
    bVariable3    : BYTE    := 3;
    dwVariable4   : DWORD   := 7;
```

```
(* Values to transfer *)
rVariable1 := rVariable1 + 1;
wVariable2 := wVariable2 + 2;
bVariable3 := bVariable3 + 3;
dwVariable4 := dwVariable4 + 4;

(* Init collection 1 *)
aVariableDescriptions1[0].pAddress := ADR(rVariable1);
```

Das Programm wurde geändert! Sollen die Änderungen geladen werden? (ONLINE CHANGE)

Figure 3-8: Compile the program with online change

Note

Important note!
After an Online Change there might be changes concerning the data on certain addresses. Please regard this in case of using pointers on addresses.

4 Commissioning

4.1 General configuration

This chapter shows a view of general information and settings for the PFC. It is strongly recommended to read this chapter.

- Please consider that the enabled Cloud Connectivity can decrease the performance of the PFC.
- PFC200: Please keep in mind that you have to configure the runtime. You have the choice between Codesys 2.3 and e!RUNTIME. The runtime can be configured by using the WBM.
    1. Launch the WBM in a Web browser.
    2. Click on the **PLC Runtime** menu item
    3. Enter the **Username** and **Password** to authenticate on the PFC. Click [**Submit**]. A message box may appear that prompts you to change the password for security reasons.
    4. Click [**Submit**]
    5. General PLC Runtime Configuration will be shown.
    6. If the Codesys 2.3. library should be used than select "Codesys 2" for **PLC runtime version** otherwise select "e!RUNTIME".

Note

Change passwords
Default passwords are documented in these instructions and therefore do not offer adequate protection! Change the passwords to meet your particular needs for several groups (WBM: admin, user – Linux: root, admin, user). For further information see into the manual of the PFC.

4.1.1 Establishing an Internet Connection to the PFC

The PFC is to be connected to the Internet so that:

- A cloud service can be set up on the Internet.
- The NTP client can synchronize with a time server.

The PFC200 (750-8207) has a mobile radio modem that can be used to connect to the Internet.

All other PFCs require an external mobile radio modem that can be connected using one of two ETHERNET interfaces or an Internet connection via network cable to one of two ETHERNET interfaces.

Assumption: Both the X1 and X2 ETHERNET interfaces obtain their IP addresses via DHCP.

The configuration example described below for a mobile radio modem that is connected externally assumes that the PFC ETHERNET interface:

- DHCP is configured on the Configuration Type.
- An IP address for the PFC is obtained from the DHCP server via the mobile radio modem.
- An IP address for a gateway is obtained from the DHCP server via the mobile radio modem.
- An IP address for the DNS server is obtained from the DHCP server via the mobile radio modem.

Configuration example:

1. Launch the WBM in a Web browser.
2. Move the mouse over the **Networking** menu item and select the **ETHERNET** menu item.
3. Select "Separated" in the Switch Configuration section and click [**Submit**] to write the parameter.

4. Move the mouse over the **Networking** menu item and select the **TCP/IP** menu item.
5. Check that the **DHCP** configuration type is selected for both ETHERNET interfaces. If not, change the settings and click [**Submit**] to save.

Information

Additional information
It is also possible to assign a static IP address to one or both ETHERNET interfaces. If the ETHERNET interface by which the Internet connection is made is configured using a static IP address, make sure that a gateway is specified and enabled in the **TCP/IP** menu item. Otherwise, the Internet connection will not work.

6. Log into the PFC via “Remote Shell”.
7. After logging into the PFC successfully via “Remote Shell”, the ping 8.8.8.8 command can be executed.
8. Enter nslookup www.google.de. If the IP address of www.google.de is returned, name resolution works, too.
9. If one or both tests fail, steps 1 - 9 should be checked.

4.1.2 Ports Used for Cloud Connectivity

The Cloud Connectivity functionality uses ports 1883 (unencrypted connection) and 8883 (encrypted connection). To ensure communication between the Cloud functionality on the PFC and the cloud service, make sure that the specified ports are Internet accessible and not blocked by a firewall for incoming data traffic. In other words, if incoming data traffic is blocked on the above ports, no connection between the Cloud Connectivity functionality on the PFC and the cloud service is possible.

Via „Remote Shell“ it is possible to verify whether cloud service is connectable from your PFC, i.e.: *telnet wagocloud.azure-devices.net 8883*. If the port port is accessible after some minutes the host closed the connection and give back the follow message: *Connection closed by foreign host*. If the port is blocked by a firewall the follow message is coming: *telnet: can't connect to remote host (13.69.192.43): Connection timed out*.

4.1.3 Internet Connection via a Web Proxy Server

To use the Cloud Connectivity functionality, an Internet connection to a cloud service configured in WBM is required. The Internet connection must be direct, i.e., an Internet connection via an intermediate Web proxy server is not possible.

4.1.4 Activate Time Synchronsiation

It is imperative to ensure that the PFC's time and date are correct. If the time and date are incorrect, problems arise when checking certificates. As a result the secure, certificate based network connection to the cloud service would always fail. Therefore, it is recommended that you enable the PFC's NTP client.

The NTP client will be activated with WBM:

1. Launch the WBM in a Web browser.
2. Click the Ports and Services menu item.
3. Enter the **Username** and **Password** to authenticate on the PFC. Click [**Submit**]. A message box may appear that prompts you to change the password for security reasons.
4. Click the NTP Client menu item.
5. Choose the button Service enabled and enter the time server address. (e.g. the google NTP server: 216.239.35.8 or the time server of the ‘Physikalisch-Technische Bundesanstalt’ in Braunschweig: 192.53.103.108).
6. Click the **[Submit]** button to take the settings.
7. Control the time.

Note

Important note!
If the clock on the PFC is wrong, then it is impossible to connect to the Cloud. This is because the handshake for establishing the encrypted TLS connection is based on the correct date and time.

For further information see into the manual of the PFC.

4.2 Setup Cloud Connectivity

This chapter describes as a quick start guide how to set up a dedicated cloud service and configure the Cloud Connectivity functionality for it.

The cloud services are configured on the Cloud Connectivity page in WBM. Input masks for the following cloud services are available:

- WAGO Cloud
- Azure
- IBM Cloud
- Amazon Web Services (AWS)
- SAP IoT Services

A special form takes the “MQTT AnyCloud” entry. A connection to any cloud can be established that supports MQTT Protocol, Version v3.1/v3.1.1 such as the MQTT broker on the Mosquitto stack.

4.2.1 WAGO Cloud

The steps required to connect to WAGO Cloud are described below. A more detailed description of the WAGO Cloud user interface is included in the WAGO Cloud Quickstart Reference.

4.2.1.1 Configuring Cloud Connectivity for WAGO Cloud

1. Launch the PFC's WBM in a Web browser.
2. Click the **Cloud Connectivity** menu item.
3. Enter the **Username** and **Password** to authenticate on the PFC. Click [**Submit**]. A message box may appear that prompts you to change the password for security reasons. The Cloud Connectivity configuration page opens.
4. In the “Settings” section, select the “WAGO Cloud” entry in the **Cloud Platform** selection box.
5. Default **Host Name** is “wagocloud.azure-devices.net”, can be changed if required so.
6. Enter the device ID in the **Device ID** field. This is the device ID from the WAGO Cloud.
7. Enter the activation key from WAGO Cloud in Activation Key.
8. The **Cache Mode** dropdown box in the “Operation behavior” section can be used to select whether data is cached in RAM (volatile) or on the SD card (non-volatile).
9. Click the [**Submit**] button to write the configuration.
10. The PFC must be restarted to apply the settings.

4.2.1.2 PLC Application for WAGO Cloud

The PLC program developer is responsible for observing the naming conventions for tags and collections. If this convention is disregarded, no data will be displayed in the WAGO Cloud.

The following rules apply for strings used for the **Tag** of a variable

- It must be unique within a collection
- It must follow rules for C# identifiers
- Following characters are not allowed for the string used for the Tag of a variable
    - The “space” character
    - The forward slash (/) character
    - The backslash (\) character
    - The number sign (#) character
    - The question mark (?) character
    - Dash (-)
    - Control characters including i.e. the horizontal tab (\t) character, the linefeed (\n) character and the carriage return (\r) character

Following characters are not allowed for the string used for the name of a **collection**:

- The forward slash (/) character
- The backslash (\) character
- The number sign (#) character
- The question mark (?) character
- Control characters, including i.e. the horizontal tab (\t) character , the linefeed (\n) character and the carriage return (\r) character

4.2.2 Microsoft Azure

Azure account is required in order to use Azure Cloud Services. In case you want use Azure Services for the first time you might start by creating a free Azure account (https://azure.microsoft.com/en-us/free/). Such Azure account is limited by credit limit and time span. Afterwards it only becomes liable to pay when you decide for further subscription.

Having an Azure account check the following subchapters describing how to setup the needed services using Azure Portal.

4.2.2.1 Setting up Azure IoT Hub

Azure IoT Hub is a fully managed service that enables reliable and secure bidirectional communications between millions of IoT devices and a solution back end. It is the communication endpoint to Azure from the view of your Wago PFC.

To create IoT Hub please do these steps:

1. Sign in to the Azure Portal https://portal.azure.com
2. In the Jumpbar, nagigate to New > Internet of Things > IoT Hub

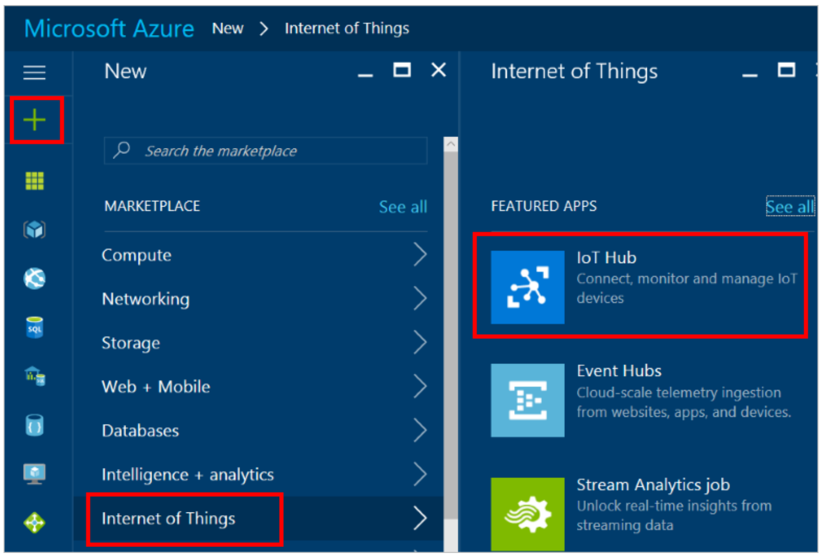
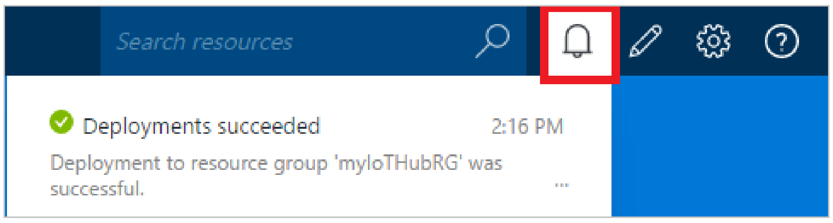


Figure 4-1: Add IoT Hub

3. Enter name, select pricing and scale tier, create new resource group and select the location for your new IoT Hub (please check online documentation for details https://docs.microsoft.com/en-us/azure/iot-hub/quickstart-send-telemetry-dotnet)
4. It can take a few minutes for Azure to create your new IoT Hub. To check the status, you can monitor the progress on the Startboard or in the Notifications panel.



Deployments succeeded 2:16 PM

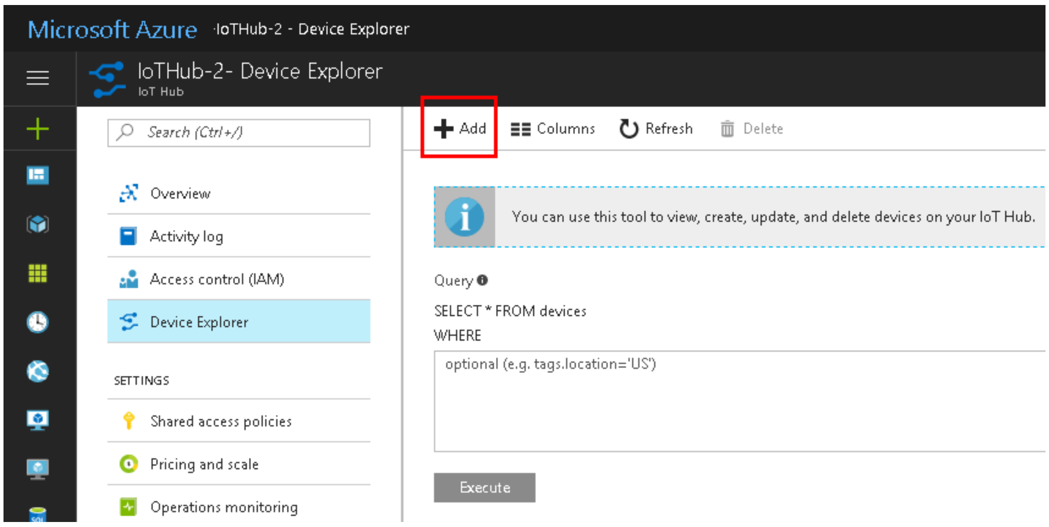
Deployment to resource group 'myIoTHubRG' was successful.

Figure 4-2: Status of deployment

4.2.2.2 Registering new device

To register new device to your IoT Hub please do these steps:

1. Within Azure Portal navigate to Device Explorer of your IoT Hub and click **[Add]**

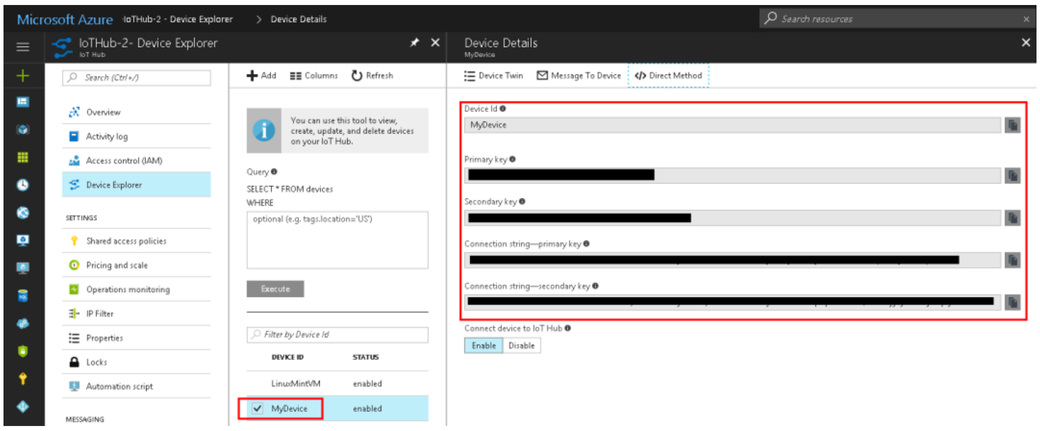


You can use this tool to view, create, update, and delete devices on your IoT Hub.

Query
SELECT * FROM devices
WHERE
optional (e.g. tags.location='US')

Figure 4-3: Add device

2. Enter ID for your device, select symmetric key, let auto-generate keys and finally press **[Save]** button
3. Added device will be listed. When clicking it you will see all the details including its keys and connection strings.



You can use this tool to view, create, update, and delete devices on your IoT Hub.

Query
SELECT * FROM devices
WHERE
optional (e.g. tags.location='US')

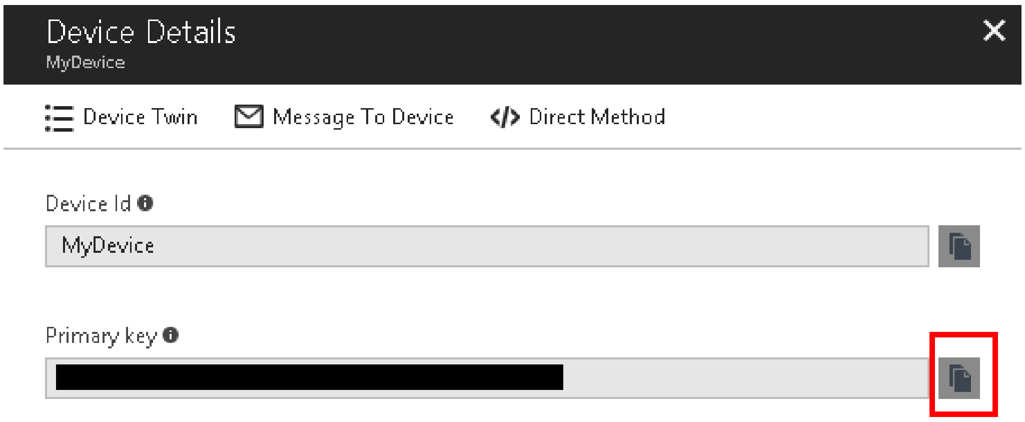
DEVICE ID	STATUS
LinuxMintVM	enabled
✓ MyDevice	enabled

Device Id: MyDevice
Primary key: [redacted]
Secondary key: [redacted]
Connection string—primary key: [redacted]
Connection string—secondary key: [redacted]
Connect device to IoT Hub

Figure 4-4: Device details

4.2.2.3 Configuring Cloud Connectivity

1. Within Azure Portal select your device and copy the primary key to clipboard

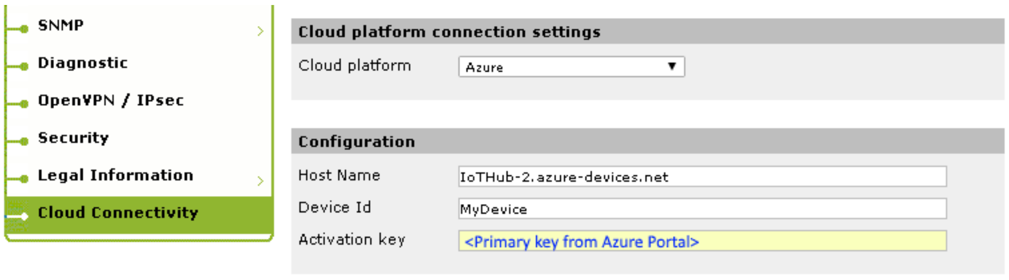


Device Id: MyDevice

Primary key: [redacted]

Figure 4-5: Device primary key

2. Launch WBM of your Wago PFC and navigate to Cloud Connectivity
3. Select Azure as cloud platform
4. Enter hostname of your IoT Hub and Device Id respectively
5. Paste primary key from clipboard into field named Activation key



Cloud platform: Azure

Host Name: IoTHub-2.azure-devices.net
Device Id: MyDevice
Activation key: <Primary key from Azure Portal>

Figure 4-6: Configuration Cloud Connectivity

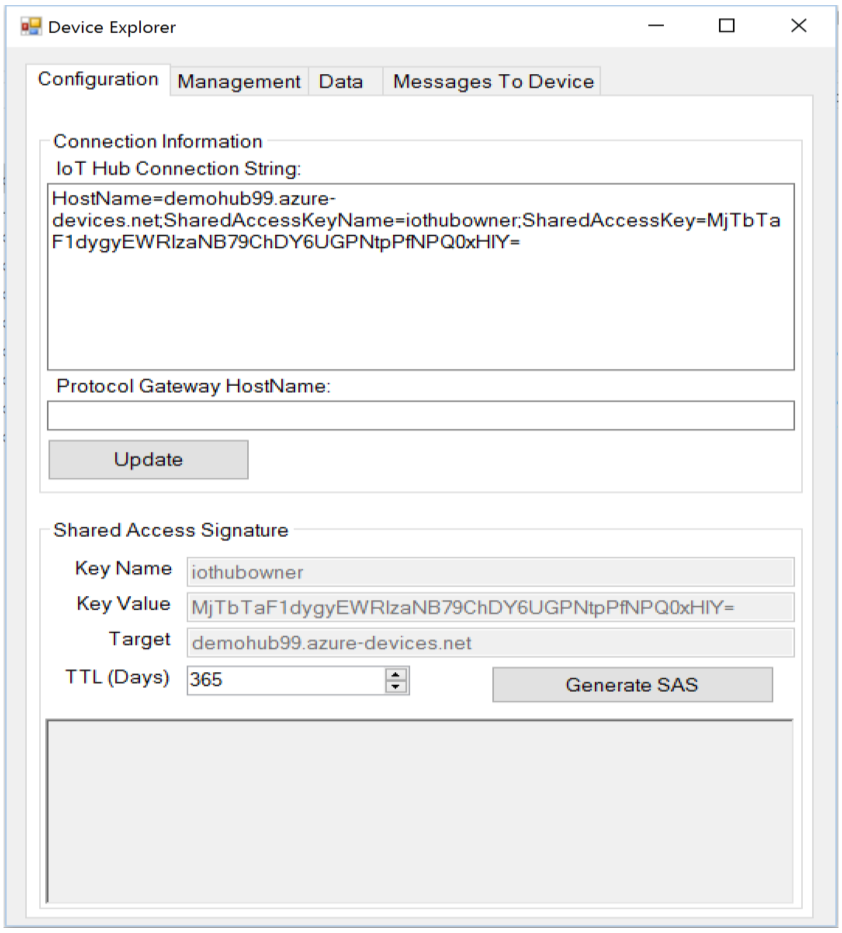
5. Click **[Submit]** to save your configuration and reboot your Wago PFC
6. Status area indicates whether your Wago PFC is connected toyour Azure IoT Hub

4.2.2.4 Monitoring device-to-cloud messages

Azure Portal shows only the total number of messages received by IoT Hub, so that indication is not very helpful during development phase.

This topic describes how to use Device Explorer Tool to view the messages which are sent by your Wago PFC to your Azure IoT Hub.

1. Make sure your Wago PFC is connected to your Azure IoT Hub and your PLC program is continuously forwarding data to the cloud platform
2. Download a pre-built version of the Device Explorer Tool from https://github.com/Azure/azure-iot-sdk-csharp/releases, install and run it
3. Make sure port 5671 is open because Device Explorer Tool uses it to retrieve data from IoT Hub
4. In the Configuration tab enter/paste connection string of your IoT Hub and click **[Update]**



Device Explorer

Configuration | Management | Data | Messages To Device

Connection Information

IoT Hub Connection String:

HostName=demohub99.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=MjTbTaF1dygyEWRlzaNB79ChDY6UGPNtpPfNPQ0xHIY=

Protocol Gateway HostName:

Update

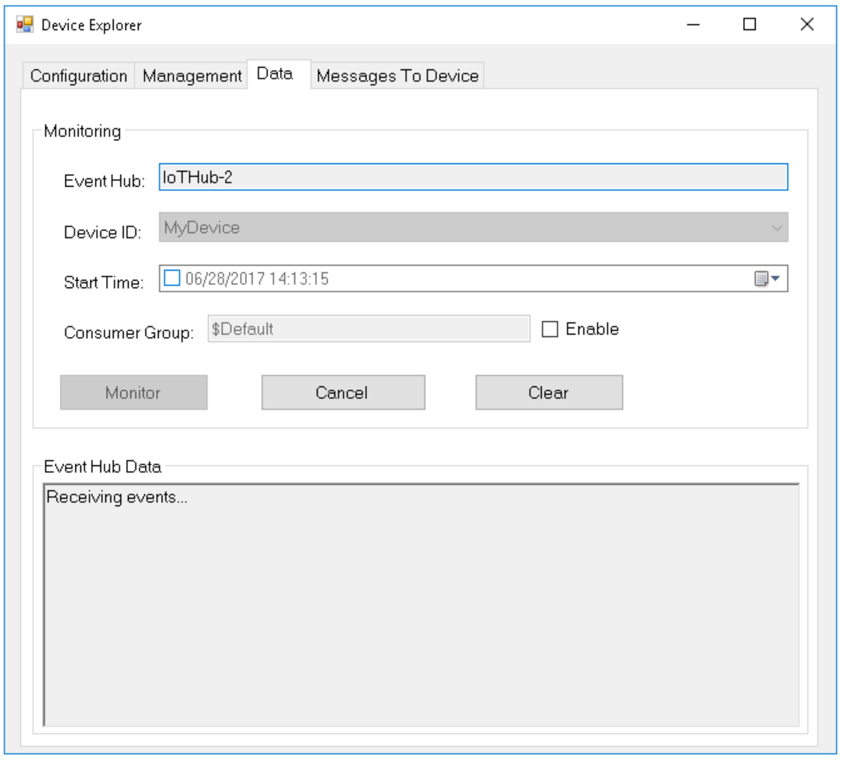
Shared Access Signature

Key Name	iothubowner
Key Value	MjTbTaF1dygyEWRlzaNB79ChDY6UGPNtpPfNPQ0xHIY=
Target	demohub99.azure-devices.net
TTL (Days)	365

Generate SAS

Figure 4-7: Device Explorer Tool

5. In the tab Data select your device and click **[Monitor]** button



Device Explorer

Configuration | Management | Data | Messages To Device

Monitoring

Event Hub:	IoTHub-2
Device ID:	MyDevice
Start Time:	☐ 06/28/2017 14:13:15
Consumer Group:	$Default ☐ Enable

Monitor | Cancel | Clear

Event Hub Data

Receiving events...

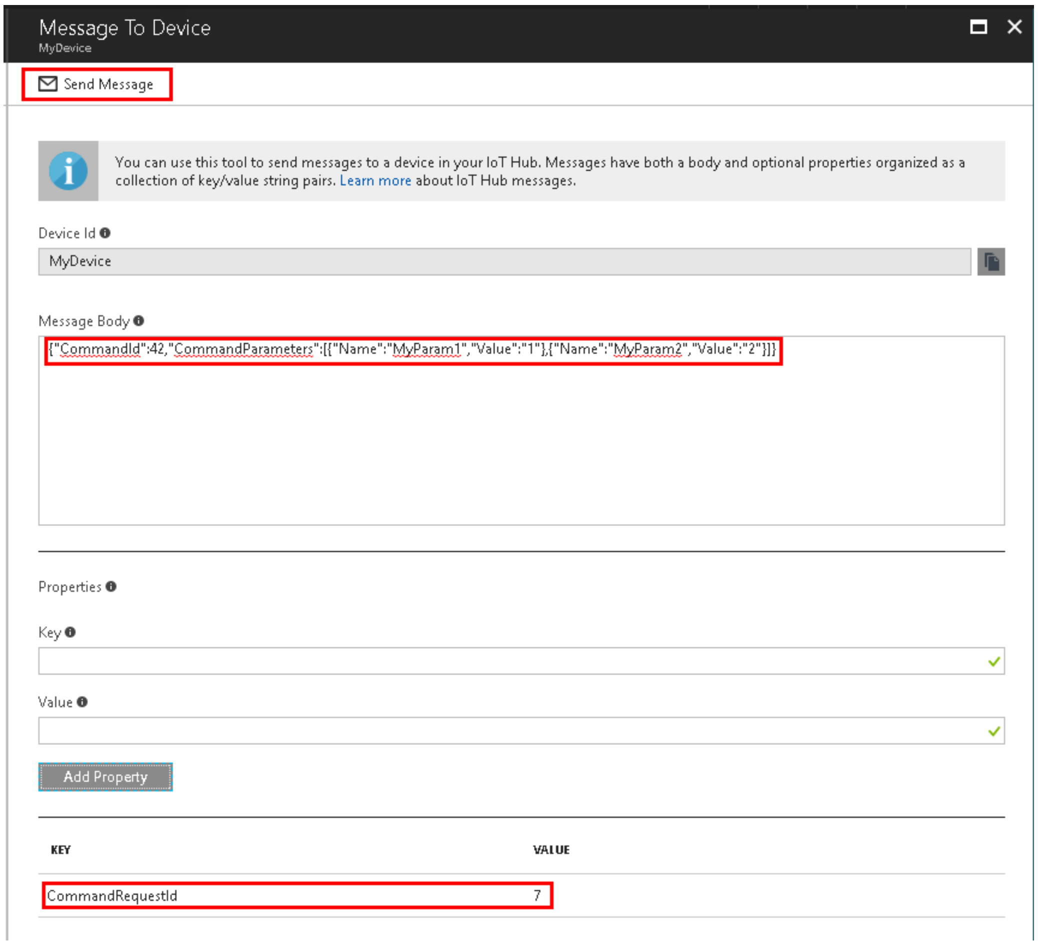
Figure 4-8: Monitoring device-to-cloud messages

7. When your IoT Hub has received message from your Wago PFC then Device Explorer Tool will request and display it promptly

4.2.2.5 Send cloud-to-device messages

It is possible to send commands from Azure to your Wago PFC using Device Explorer Tool or Azure Portal. This chapter describes the way to do it via Azure Portal.

1. Make sure your Wago PFC has network connection to Azure
2. Make sure your Wago Controller can execute the command yout want send to it
3. In Azure Portal navigate to Your IoT Hub > Device Explorer > Your Device > Message To Device
4. Enter your command as Message Body, add property named CommandRequestId with any number as value, click **[Send Message]**



You can use this tool to send messages to a device in your IoT Hub. Messages have both a body and optional properties organized as a collection of key/value string pairs. Learn more about IoT Hub messages.

Device Id: MyDevice

Message Body:

```
{"CommandId":42,"CommandParameters":[{"Name":"MyParam1","Value":"1"},{"Name":"MyParam2","Value":"2"}]}
```

Properties

Key

Value

KEY	VALUE
CommandRequestId	7

Figure 4-9: messages from the cloud to the device

5. Observe that your Wago PFC has executed the command you sent in previous step

4.2.3 Amazon Web Services (AWS)

The steps required to use the IoT service from AWS are described below.

First of all an account must be created at https://aws.amazon.com. There is no cost to create an account. You can start a test phase in which 250,000 messages/month can be sent in the first year at no cost.

4.2.3.1 Sign in to the AWS IoT Service

After successfully login in to the AWS Management Console, open the AWS IoT Console (Service: IoT Core)

Please refer to the AWS doumentation for more information:
https://docs.aws.amazon.com/iot/latest/developerguide/iot-console-signin.html

You can choose where service data should be hosted or on which servers the service should run. Use the selection box at the top right next to the "Service" selection box.

Information

Additional information
Devices created later and messages send to the cloud are only visible if you select the right server location. "Right" means that a device with the setting N. Virginia has been created. The created device and messages send to the cloud are then also visible when N. Virginia is selected.

4.2.3.2 Creating a new device in the AWS IoT Service

To successfully create a device at the AWS IoT console, the following steps must be performed:

1. **Register a device in the Registry**
   Select *Manage* in the left navigation pane and create a new device.
   (https://docs.aws.amazon.com/iot/latest/developerguide/register-device.html)
2. **Create and Activate a Device Certificate**
   Select *Secure - Certificate* in the left navigation pane and create a new certificate.
   (https://docs.aws.amazon.com/iot/latest/developerguide/create-device-certificate.html)

Note

Important note!
Download the created certificates because later the certificates must be copied to the PFC. You need the certificate for this device (.cert) and the private key.

3. **Create an AWS IoT Policy**
   Select *Secure – Policy* in the left navigation pane and create a new policy.
   (https://docs.aws.amazon.com/iot/latest/developerguide/create-iot-policy.html)
4. **Attach an AWS IoT Policy to the Device Certificate**
   Select *Secure - Certificate* in the left navigation pane and attach the policy to the certificate.
   (https://docs.aws.amazon.com/iot/latest/developerguide/attach-policy-to-certificate.html)
5. **Attach a Certificate to a device**
   Select *Secure - Certificate* in the left navigation pane and attach the certificate to the device.
   (https://docs.aws.amazon.com/iot/latest/developerguide/attach-cert-thing.html)

4.2.3.3 Configuring Cloud Connectivity for Amazon Web Services (AWS)

This section describes the steps to configure the Cloud Connectivity functionality of the PFC for Amazon Web Services (AWS). It is assumed that the AWS IoT service and a device have been set up in advance.

The following list describes the required steps to connect Cloud Connectivity to AWS.

1. Launch the PFC's WBM in a Web browser.
2. Click the **Cloud Connectivity** menu item.
3. Enter the **Username** and **Password** to authenticate on the PFC. Click [**Submit**]. A message box may appear that prompts you to change the password for security reasons. The Cloud Connectivity configuration page opens.
4. In the “Settings” section, select the “Amazon Web Services (AWS)” entry in the **Cloud Platform** selection box.
5. Enter the Amazon host name for the “AWS IoT Service” in the **Host Name** field. For the host name you have to find your personal endpoint

   To find the custom endpoint for the device, the following steps are necessary:

   - Select *Manage – Things* in the left navigation pane.
   - Choose your device.
   - Select *Interact* in the left navigation pane.
   - The Rest API Endpoint is in the area HTTPS. This is your host name.

   The host name looks something like this:

   ABCDEFG1234567-ats.iot.us-east-1.amazonaws.com

   Here ABCDEFG1234567 is the subdomain (own AWS account) and us-east-1 the region.
6. Enter the device ID in the **Device ID** field. The device ID value can be freely selected, but should not contain any special characters.
7. The **CA File** field is used to specify the path to the CA file. Enter */etc/ssl/certs/ ca-certificates.crt* here.
8. Then upload the certificate called “Name_of_the_device.cert.pem” from the file downloaded by AWS to the /etc/ssl/certs/ folder on the PFC.[2]

Information

Additional information

Name_of_the_Device is the name entered when registering the device in the AWS IoT Service.

9. Enter the path with the file name of the certificate uploaded in step 8 in the **Certificate File** field.
10. Then upload the private key called “Name_of_the_device.private.key” from the file downloaded by AWS to the /etc/ssl/certs/ folder on the PFC.
11. Enter the path with the file name of the private key uploaded in step 10 in the **Key File** field.
12. The following three settings are possible in the “Operation Behavior” section.
    1. The **Cache Mode** checkbox can be used to set whether data is cached in RAM (volatile) or on the SD card (non-volatile) if the connection is interrupted. When the connection is restored, the data is transferred to the cloud. The cache memory is limited to 3 MiB in RAM and 512 MiB on the microSD card.
    2. The **DeviceInfo** checkbox is used to enable or disable transmission of device info.
    3. The **Device Status** checkbox is used to enable or disable transmission of the device status.

15. Click the [**Submit**] button to write the configuration.
16. The PFC must be restarted to apply the settings.

[2] The WinScp tool can be used in Microsoft Windows for this purpose. The scp command line tool can be used in Linux®.

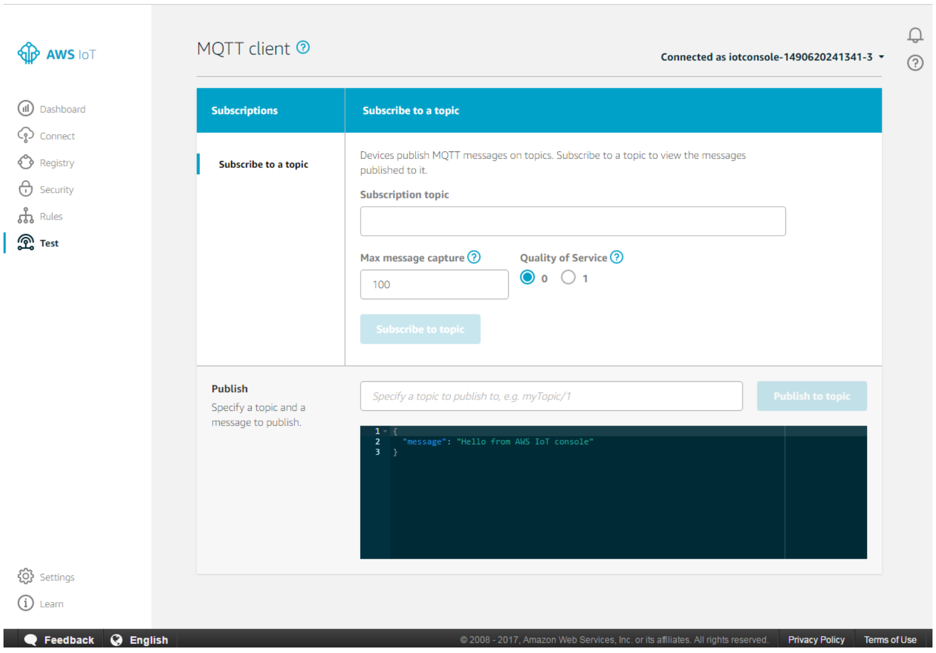
4.2.3.4 Testing a Created Device

You can test a device previously created from the **Test** menu item. Before the test can be successfully completed, the PFC must be configured for the AWS IoT service. Please refer to Section 4.2.3.3.

Information

Additional information
The device to be tested must be in the namespace currently selected because the device and messages are otherwise not visible. In other words, a device created in the N. Virginia namespace is also only visible there. This also applies to the data.

1. Click the **Test** menu item. The MQTT client opens. (https://docs.aws.amazon.com/iot/latest/developerguide/view-mqtt-messages.html)



Devices publish MQTT messages on topics. Subscribe to a topic to view the messages published to it.

Max message capture: 100
Quality of Service: [✓ 0] 1

Specify a topic and a message to publish.

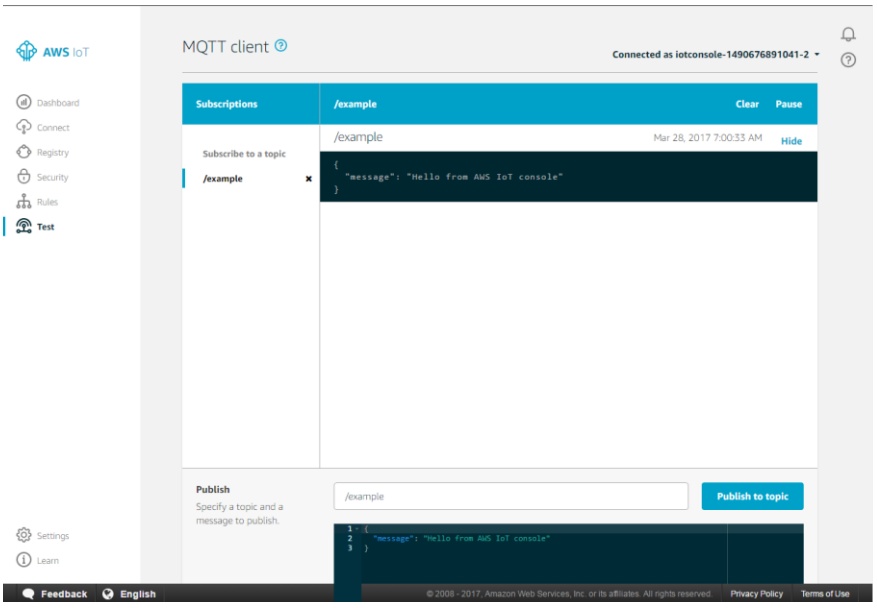
Specify a topic to publish to, e.g. myTopic/1

```
{
  "message": "Hello from AWS IoT console"
}
```

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Figure 4-10: Amazon MQTT Client

2. Enter the value “Device_ID_from_the_WBM/#” in the **Subscription topic** field. Device_ID_from_the_WBM is the field value from the WBM Cloud Connectivity site.
3. Click [**Subscribe to topic**]. The subscription is displayed on the left side.
4. Click the subscription you just created. An empty window opens on the right next to the subscription.



```
{
  "message": "Hello from AWS IoT console"
}
```

```
{
  "message": "Hello from AWS IoT console"
}
```

Figure 4-11: MQTT Client, Subscription View

5. To publish, the topic is automatically applied from the subscription. If not, enter the topic in the field to the right of “Publish”.
6. Click [**Publish to topic**] to send the message in the small console window below the Topic field to the “AWS IoT MQTT Broker”.
7. Check if the sent message appears at the top in the MQTT client under Subscriptions. If yes, the configuration is working.

4.2.4 IBM Cloud

The steps required to use the “Watson IoT Platform” service in IBM Cloud are described below.

4.2.4.1 Creating a User Account and Service in IBM

1. Create test account (valid only 30 days, no credit card required)
2. Login (https://login.ng.bluemix.net)
3. In the IBM Cloud, click the **Create Resource** button under Dashboards.
4. Select **Internet of Things Platform** from the left menu.
5. Select an appropriate pricing structure (try the free “Lite” plan) and build the IBM Watson IoT Platform.
6. The “Internet of Things Platform” service details then appear.
7. Click [**Start**] to display the “IBM Watson IoT Platform Service”.
8. In the menu list on the left side, click the **Devices** entry for device management.
9. Click [**Add Device**] to add a device.
10. Then create a device type by clicking [**Create Device Type**]. You are asked whether you want to create a “Device type” or “Gateway type”. Select “Device type”. Enter the name and description of the “Device type” in the next dialog. Define a template (optional) in the next dialog. Click [**Next**] to go to the next dialog. If you have selected fields for the template, the fields defined in the template are to be filled out. Click [**Next**] to go to enter the optional “metadata”. Click [**Create**] to create the device type.
11. The previously created device type is then selected. By clicking [**Next**], you are prompted in the next dialog to specify a **device ID** (required). The fields selected from the template of the device type previously created also appear. The fields are automatically populated with default values from the template and are to be adjusted accordingly. In the next step, you are prompted for the optional metadata that must be added in JSON format. Click [**Next**] to select the security to be used. The following options are available: “Automatically generated authentication token” or “Self-provisioned authentication token”. Click [**Next**] to select automatic generation of the token. The next dialog displays a summary of all device-related data. Click [**Add**] to create a device and to display the identification details for the device. This dialog should be saved, so that you have the device data at hand to register the device.

4.2.4.2 Configuring Cloud Connectivity for IBM Cloud

This section describes the steps to configure the Cloud Connectivity functionality of the PFC for IBM Cloud. It is assumed that the IBM Cloud service and a device have been set up at IBM in advance. It is also assumed that the identification data for a device is available.

The following list describes the required steps to connect Cloud Connectivity to IBM Cloud.

1. Launch the PFC's WBM in a Web browser.
2. Click the **Cloud Connectivity** menu item.
3. Enter the **Username** and **Password** to authenticate on the PFC. Click [**Submit**]. A message box may appear that prompts you to change the password for security reasons. The Cloud Connectivity configuration page opens.
4. In the “Settings” section, select the “IBM Cloud” entry in the **Cloud Platform** selection box.
5. Enter the IBM host name for the “IBM Watson IoT Platform” in the **Host Name** field. The host name is configured as follows:

   org_id.messaging.internetofthings.ibmcloud.com
   org_id must be replaced by the respective organization ID.

Information

Additional information
The organization ID was specified when creating a device in “Your Identification Data for the Device”.

6. Enter the device ID in the **Device ID** field. The structure of the device ID is shown below:
   d:*org_id*:*device_type*:*device_id*

Table 1: Explanation of the Device ID Substrings

d	The “d” signals to the “IBM Watson IoT Platform Service” that the string is a device.
org_id	To be replaced by its own organization ID.
device_type	The device type is defined by the DeviceType that has been assigned to the device in the “IBM Watson IoT Platform Service”.
device_id	The device ID assigned to the device

7. The **TLS** checkbox can be used select whether the connection is encrypted and data is transferred in encrypted format.
8. The **CA File** field is used to specify the path to the CA file. Enter */etc/ssl/certs/ca-certificates.crt* here.
9. Select the value “1883” in the **Port** field for an unencrypted connection. Select **Port** “8883” for an encrypted connection.
10. Enter the value “use-token-auth” in the **User** field.
11. Enter the password in the **Password** field (authentication token) from the device settings in the IBM Cloud.
12. The following three settings are possible in the “Operation Behavior” section.
    1. The **Cache Mode** checkbox can be used to set whether data is cached in RAM (volatile) or on the SD card (non-volatile) if the connection is interrupted. When the connection is restored, the data is transferred to the cloud. The cache memory is limited to 3 MiB in RAM and 512 MiB on the microSD card.
    2. The **Device Info** checkbox is used to enable or disable transmission of device info.
    3. The **Device Status** checkbox is used to enable or disable transmission of the device status.

15. Click the [**Submit**] button to write the configuration.
16. The PFC must be restarted to apply the settings.

4.2.4.3 Creating Data Visualization

The data sent to IBM Cloud can be displayed in the “IBM Watson IoT Platform” in graphical or textual form. Various display types are available. As an example, some data from the “DeviceInfo” structure is displayed in a so-called card.

The dashboard with available boards serves as a starting point. In the Web browser, we are going to the following site https://*org_id*.internetofthings.ibmcloud.com/dashboard/boards/
org_id is to be replaced by its own organization ID.
An overview of all boards appears after logging in (see Figure 4-12).

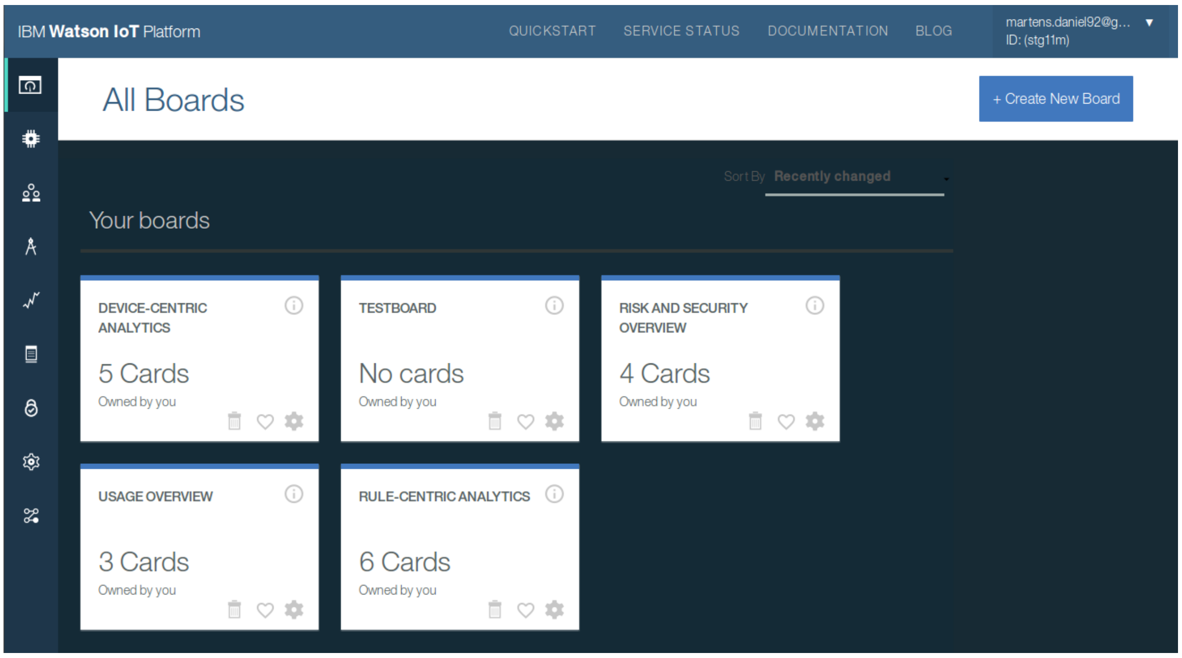


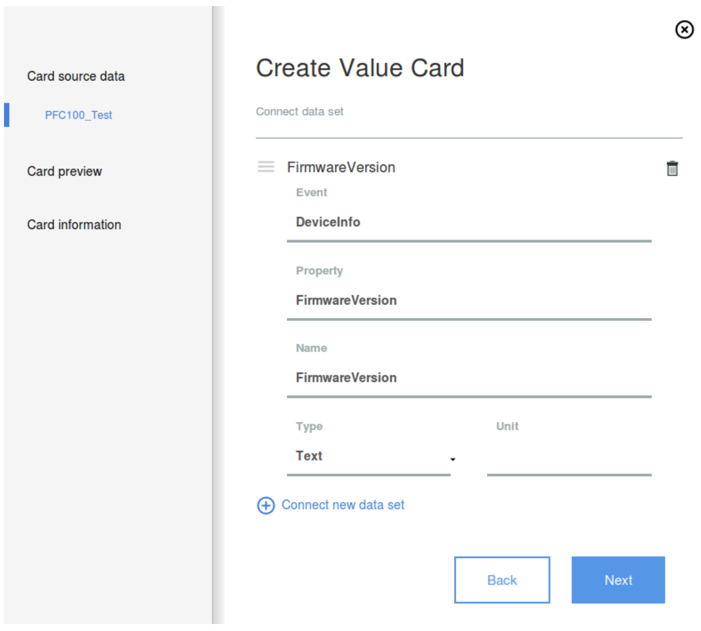
Figure 4-12: Overview of All Boards

We will now create a new board with a new card:

1. Click [+**Create New Board**]. The dialog for creating boards appears. Enter the “Name” and a “Description” for the board.
2. Click [**Next**] to display other board settings. Click [**Apply**] to apply the settings. The new board is displayed in the board overview after a few seconds.
3. Then click “Board” to put the focus on the previously created board. You can then change the board settings or add a new card.
4. Click [**+Add New Card**] to open a dialog for selecting a card type.
5. For example, select the device “value” by clicking on it.
6. All available devices are then displayed. Select the device that should serve as the source. Click [**Next**] to display the dataset settings.
7. Click [**Connect to New Dataset**] to display other fields. Click the field below “Event” to display a list of the events previously delivered to IBM Cloud, e.g., “DeviceInfo”. [3] We select them.
8. Click the field below “Property” to display a list of the properties previously delivered, e.g., “Firmware Version”. We select them.
9. The “Name” field is automatically populated with the property name, but can also be renamed.
10. The “Type” field specifies the property types, e.g., “Text” or “Number” can be selected. In this example, select “Text”.

Information

Additional information
More than one service can be created on one card (see Figure 4-13)



FirmwareVersion
Event: DeviceInfo
Property: FirmwareVersion
Name: FirmwareVersion
Type: Text
Unit:

Figure 4-13: Selecting Data for the Card

11. Click [**Next**] to display a card preview (see Figure 4-14).

[3] So you can select events here, data must have been sent to the cloud in advance. If no data has been sent to the cloud, the selection list is empty.

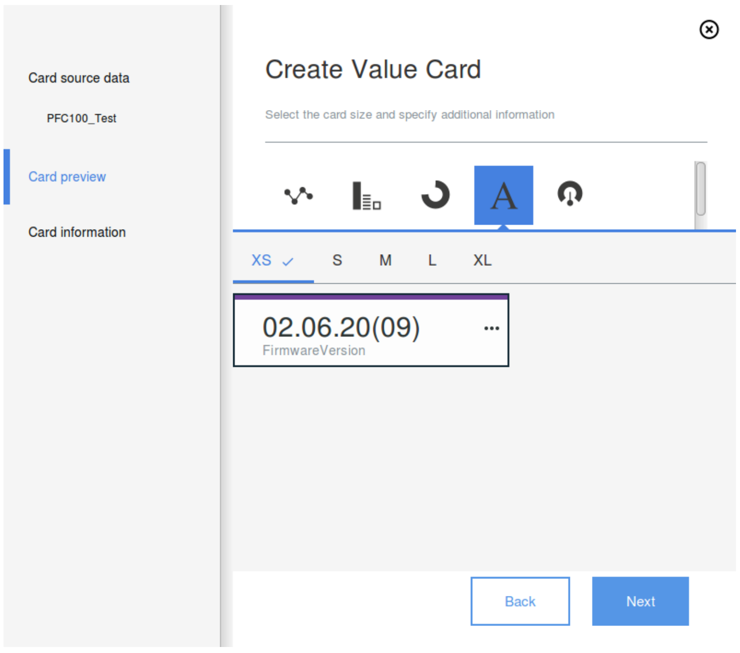


Figure 4-14: Preview of the Card with Selected Data

11. Click [**Next**] to display a summary and to select a color scheme.
12. Click [**Submit**] to complete the card setup.

4.2.4.4 Customer Applications

- General Information on IBM Cloud Services
  - https://www.ibm.com/cloud-computing/bluemix/de/watson
- Getting Started
  - https://console.ng.bluemix.net/docs/services/IoT/index.html

4.2.5 SAP IoT Services

The steps for using the SAP IoT Services are described below. Your SAP consultant creates an account in SAP for you.

4.2.5.1 SAP IoT Service Cockpit: Log On

You log on to the SAP Demo IoT Service using the URL:
https://com-iotaedemo.eu10.cp.iot.sap You will receive the final link and access data from your SAP consultant.

After registration you will be taken to the "Internet of Things Service Cockpit". **Create** the desired users in your **Tenant** under **User Managements** > **User** (see Create User and Tenant). Your SAP consultant can help you to create the **Tenant.**

4.2.5.2 SAP IoT Service Cockpit: Create a Device

By selecting the **Tenant** the **Device Management** becomes visible in the menu.

1. Select **Device Management > Devices** from the menu and click **[+].**
2. Fill in the mandatory field **Name**.
3. Configure **Gateway** as **[MQTT Network (ID: 2)].**
4. Complete the creation of the new Device with **[Create]** and **[Confirm]**.

The Device is now created. In the next steps, the data structures supported by the device are defined.

4.2.5.3 SAP IoT Service Cockpit: Create a Capability

A capability is a template of a data structure that is transferred from the WAGO device to the SAP IoT service.

1. Select **Device Management > Capabilities** from the menu and click **[+]**.
2. Fill in the mandatory field **Name**.
3. Create a list of the variables to be transferred in the **Properties** table. Enter a **Name** for each variable and select the **Data Type**. Optionally, you can also enter a **Unit of Measure** for each variable. To add more variables, click **[+]**.
4. Finish creating the new capability with **[Create]** and **[Confirm]**.

4.2.5.4 SAP IoT Service Cockpit: Create a Sensor Type

A sensor type is a template that contains at least one capability.

1. Select **Device Management** > **Sensor Types** from the menu and click **[+].**
2. Fill in the mandatory field **Name**.
3. Select the capabilities you want to support from the list of **Capabilities** and assign the type to each one in the **Type** field**.** Possible values for **Type** are **measure** and **command.** The **Type measure** represents data that is published by the device. The **Type command** represents data subscribed by the device. To add more variables, click **[+]**.
4. Complete the creation of the new sensor type with **[Create]** and **[Confirm]**.

4.2.5.5 SAP IoT Service Cockpit: Create a Sensor

1. Select your [**Device]** in the **Device Management > Devices** menu.
2. Select **[Sensors]** and click **[+]** to create a new sensor.
3. Fill in the mandatory field **Name**.
4. Select your sensor type from the list **Sensor Type.**
5. Complete the creation of the new sensor with **[Create]** and **[Confirm]**.
6. If there is still a default sensor (**sensor: 0:0:0:0)** in the list of **Sensors**, delete it by clicking on the trash can icon and confirm the deletion with **[Confirm]**.

4.2.5.6 SAP IoT Service Cockpit: Generate a Certificate

1. Select your [**Device]** in the **Device Management > Devices** menu.
2. Select **[Certificate]** and press **[Generate Certificate]**.
3. Select the certificate type **[pem]** in the selection dialog that appears and confirm the dialog with **[Generate]**.

Note

Important note!
Make a note down the displayed Secret, as it is required in the next step.

The certificate with the name: ***Name_Your_Device-device_certificate.pem*** was created and is now in the download directory of your PC.

4.2.5.7 Separating the key of a certificate

The certificate generated by SAP cannot be processed directly by the WAGO device. For this reason, the key must be separated.

Note

Important note!
To separate the key, a working installation of the open source tool openSSL-Win32 is required (for installation instructions, see Install OpenSSL).

1. Copy the certificate from the download directory to the *openssl-win32\bin* directory.
2. Open a Windows command line with administrator privileges and navigate to the directory *openssl-win32\bin.*
3. Enter the following command, where < myCertificate.pem> has to be replaced with the name of your certificate and < myKey*.key> by the* name of the target file. Confirm with **Enter:**

   openssl rsa -in "< myCertificate.pem>" -out "< myKey. key>"
4. Enter the corresponding secret and confirm with **Enter.**
5. Download the two files (the certificate and the key) to the WAGO device in the directory: */etc/ssl/certs/*

Note

Important note!
For copying the certificates to the WAGO device an FTP client such as the open source program FileZilla (see https://filezilla-project.org/) can be used.

4.2.5.8 Configuration of Cloud Connectivity

1. Use a Web browser to call the WBM of your WAGO device.
2. Select the **[Cloud Connectivity]** menu item.
3. Enter **Username** and **Password** to authenticate to the PFC. Click **[Submit]**. Possibly, a message box will appear indicating that the password should be changed for security reasons. The Cloud Connectivity configuration page opens up.
4. In the **Settings** section, select **[SAP IoT Services]** from the **Cloud platform** drop-down list.
5. Activate the **[Service enabled]** check box.
6. Enter your SAP host name in the **Host name** field. You can obtain the host name from your SAP consultant. Example for the demo host: *com-iotaedemo.eu10.cp.iot.sap*
7. In the **Client ID** field, enter the Alternate ID of your device. (Alternate ID is displayed in the SAP IoT Service Cockpit under **Device Management > Devices**)
8. Activate the **[TLS]** selection box.
9. Enter "*8883*" in the **Port** field.
10. In the **CA File** field, enter the path to the CA file:

    "*/etc/ssl/certs/ca-certificates.crt*"

6. In the **Certification file** field, enter the path (incl. file name) of your certificate (e.g. "*/etc/ssl/certs/myCertificate.pem*").
7. In the **Key file** field, enter the path (incl. file name) of your key (e.g. "*/etc/ssl/certs/myKey.key*").
8. Select **[Submit]** and reboot the WAGO device.
9. Now check the connection status in the WBM under **Cloud Connectivity**. This must be **connected** after a few minutes.

Information

Further information
The selection fields **[Clean Session]** and **[Last Will]** can be optionally set depending on the desired behavior.

Note

Important note!
If the WAGO device does not connect to the SAP IoT service, check the previous steps. Make sure that you have entered the Alternate ID as the client ID in the WBM and that the used port is not blocked by a firewall etc.

4.2.5.9 e!COCKPIT Program

To create an e!COCKPIT program for communication with the SAP IoT Service as conveniently as possible, we recommend that you use the following libraries:

WagoAppJSON: Serializing and Parsing the JSON Payload

WagoAppCloud: Publish and subscribe data using nativeMQTT

Sending data

WagoAppJSON: Use the function block Fb_JSON_Writer_01 to serialize the payload to the JSON format. The JSON-BaseFrame is composed as follows:

Note

Important note!
For better readability, line breaks were used in the following example. These must not be part of your JSON. The placeholders marked with <> must be replaced as described below. The placeholders **#parameters** are replaced during serialization by the transferred values within the function block.

```
{
"capabilityAlternateId": "<capabilityAlternateId>",
"sensorAlternateId": "<sensorAlternateId>",
"measures": [{
"<Name Property1>": "#Parameter",
"<Name Property2>": "#Parameter"}]
}
```

1. Replace < capabilityAlternateId> , with the Alternate ID of the Capability from the SAP IoT Service Cockpit.
2. Replace < sensorAlternateId> with the Alternate ID of the sensor from the SAP IoT Service Cockpit.
3. Replace <Name PropertyX> with the names of the properties supported by Capability from the SAP IoT Service Cockpit.
4. Transfer the property values to be transferred as an array to the function block Fb_JSON_Writer_01 and start the serialization using the trigger. Please note that the order of the values within the array corresponds to the corresponding properties in the JSON-BaseFrame!

WagoAppCloud: Use the function block FbPublishMQTT to publish messages using nativeMQTT. The topic consists of the following: 'measures/< deviceAlternateId>'

1. Replace < deviceAlternateId> with the Alternate ID of the device from the SAP IoT Service Cockpit.
2. Copy the payload generated by the WagoAppJSON into a byte array using MemCopySecure() and pass it to the function block FbPublishMQTT.
3. Start the Publish process using the trigger.

Receiving data

WagoAppCloud: Use the function block FbSubscribeMQTT to subscribe messages using nativeMQTT. The topic consists of the following: 'commands/< deviceAlternateId>'

1. Replace < deviceAlternateId> with the Alternate ID of the device from the SAP IoT Service Cockpit.

2. Start the subscription process using the trigger.

WagoAppJSON: Use the function block fbJSON_Parse to parse the payload received in JSON format.

1. Copy the received payload into a byte array using MemCopySecure() and pass it to the function block fbJSON_Parse.
2. Start the parse process using the trigger.
3. Extract the individual parameter values using the GetValueByPath() method of the fbJSON_Parse function block. The path results as follows: commands/<Name Property> '. Activate the process by setting the trigger.
4. Convert the string to the required data type (for example, with STRING_TO_REAL()).

4.2.5.10 Send cloud-to-device messages

You can send cloud-to-device messages using the Internet of Things Service API. Select **Useful Links > Device Management API from** the menu. Navigate to the **Devices** section and click **POST /devices/{deviceId}/commands *Sends a command***. Click **[Try it out]** to activate the input mode.

Note

Important note!
When sending data via the IoT Service API, in contrast to the e!COCKPIT program, the Device ID, the Capability ID and the Sensor ID are required and the corresponding Alternate IDs are not used.

1. In the **deviceId** field, enter the ID of your device from the SAP IoT Service Cockpit.
2. Replace the value for **capabilityId** in the **body** with the capability ID from the SAP IoT Service Cockpit.
3. Replace the key value list of the JSON object **command** in the **body** with the property names and the corresponding values.
4. Replace the value for **sensorId** in the **body** with the ID of the sensor from the SAP IoT Service Cockpit.
5. Send the cloud to device messages by clicking **[Execute]**.
6. The **code** 200 is returned as the **server response** if the transmission was successful. In case of an error, check the **code** and the **response body** for a possible error analysis.

4.2.5.11 SAP IoT Service Cockpit: Data Visualization

SAP offers a visualization option for the data in the IoT Service Cockpit under **Device Management > Devices.**

Select your Device here and scroll to the **Data Visualization** area. Here you have the possibility to select the **sensors** as well as their **capabilities** and **properties** for the visualization.

4.2.6 Cache Mode

The Linux Application writes the data it received from PLC program, into the cache. The Linux Application continuously transmits the data to the cloud. Successfully transmitted data will automatically be removed from cache. In case of a network connection failure the data is still kept within the cache. This means that data is not lost and will be retransmitted to the cloud when internet connection comes back.

If a network connection failure takes long time than Linux Application will automatically “rotate” (that means throw away oldest but keep newest) data within the cache if the cache is reaching its maximum size. This means that some data will be lost here.

4.2.6.1 RAM

The default cache mode is RAM. The maximum size of cache is limited to 3 MiB.

In case of electrical power outage all the cached data will get lost.

It is strongly recommended to use the cache mode RAM during development of the PLC program.

When changing the PLC program (i.e. during development) while the Cloud Connectivity Linux Application is running a cold reset has to be done when deploying the new PLC program. Otherwise (warm reset) changes are not reflected and may lead to a malfunction.

4.2.6.2 SD Card

If the data, which should be transmitted to the cloud, is important and should not get lost in consequence of interruption of network connection or/and electrical power outage then it is possible to cache these data on a persistent mass storage. In case of electrical power failure, the cached data will not get lost. It will be transmitted to the cloud after reboot.

Information

Additional information
Caching to the SD Card is available for the operation mode WAGO Protocol.

To setup SD Card the following steps are necessary:

1. Setup PFC for booting from internal flash:
   1.1. Boot PFC from SD-Card
   1.2. Web Based Management / Administrations – menue / Create Image // Create bootable image von active Partition (SD) // Start Copy
   1.3. Reboot PFC without SD-card

Step 2 is necessary only if the SD-Card was used for storing PFC firmware image before. If SD-Card was used as mass storage before step 2 can be skipped. Formating the SD-Card via CBM ensures that it only holds one partition (mmcblk0p1) which can be formated as ext3.

2. Prepare SD-Card for usage as mass storage:
   2.1. Insert SD-Card
   2.2. Establish SSH connection to PFC with root privileges (https://www.youtube.com/watch?v=uKopjYASjc0)
   2.3. Open WAGO Console Based Management Tool: cbm [ENTER]
   2.4. Select Mass Storage (8) [ENTER]
   2.5. Select SD Card (1) [ENTER]
   2.6. Select FAT format medium (2) auswählen [ENTER]
   2.7. Enter Volume Name MemoryCard [ENTER], [ENTER]

2.8. Press Q,Q,Q to exit WAGO Console Based Management Tool
2.9. Reboot PFC with inserted SD-Card

3. Format SD Card with ext3
   3.1. Establish SSH connection to PFC with root privileges (https://www.youtube.com/watch?v=uKopjYASjc0)
   3.2. umount /dev/mmcblk0p1 [ENTER
   3.3. mkfs.ext3 /dev/mmcblk0p1 -L MemoryCard [ENTER]
   3.4. mount /dev/mmcblk0p1 /media/MemoryCard [ENTER]
   3.5. chmod 2775 /media/MemoryCard [ENTER]
   3.6. chgrp sdcard /media/MemoryCard [ENTER]
   3.7. reboot [ENTER]
   3.8. SD-Card is prepared. Open WBM, open „Cloud Connectivity“ configuration page and configure cache-Mode „SD-Card“:

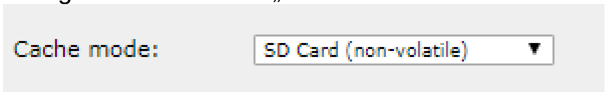

Cache mode: SD Card (non-volatile)

Figure 4-15: SD Card setup

This option is only available for operation mode WAGO Cloud and when your PFC booted from internal NAND flash, but not from memory card. It instructs the Linux Application to reserve space for cached data on mounted mass storage. By default there are following settings used:

- The cache size is limited to 512 MiB.
- Cache directory on SD Card is /media/MemoryCard

The cache mode SD Card intended to be used by a PLC program in production mode, it should not be used during PLC development.

Note

Important note!
The lifetime of SD Cards is limited. Depending on the volume of data and the sample interval the lifetime of the SD-Card are influenced. It is recommended to use the SD Card and microSD Card from WAGO. The advantages of these industrial SD Cards are the SLC storage cells (single level cell) and a special wear-leveling algorithm for an equal usage of the flash memory.

Please pay attention of the information in the data sheets of the respective manufactors.

Note

Important note!
The SD Card used shall confirm to performance class 10 or better. Otherwise, booting the PFC with cached data may take considerable time.

Note

Important note!
Since read and write operation on SD Card is slower compared to RAM the minimum sample interval shall be 1000 ms. Otherwise some data might be lost.

Note

Important note!
The cache mode SD Card should not be used during development of a PLC Program. If the PLC program changes (amount of collections or collection IDs) this may lead to data loss or space occupation on SD card beyond the configured limit.

Note

Important note!
When using the SD Card to cache data then the PLC runtime will start delayed (up to 140 seconds) after reboot because persistent data has to be analyzed first.

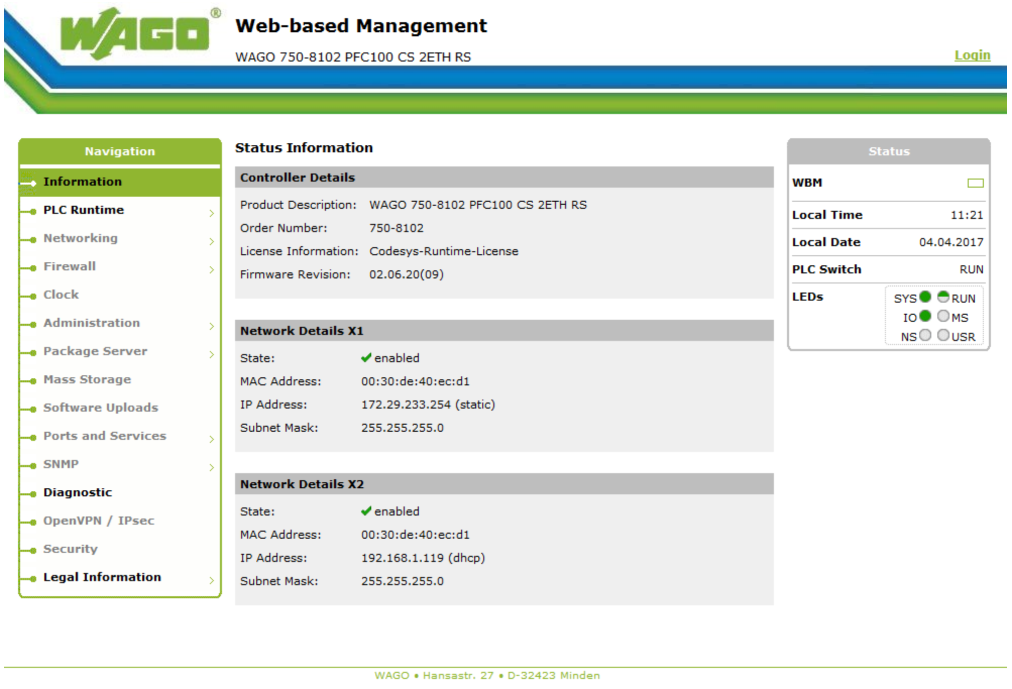
5 Appendix

5.1 Update a new version

5.1.1 Installing Linux Application

This section describes installation of the Linux Application

1. Launching the WBM in a Web browser. The WBM page shown in Figure 5-1 should be displayed.



Status Information

Controller Details

Product Description:	WAGO 750-8102 PFC100 CS 2ETH RS
Order Number:	750-8102
License Information:	Codesys-Runtime-License
Firmware Revision:	02.06.20(09)

Network Details X1

State:	✓ enabled
MAC Address:	00:30:de:40:ec:d1
IP Address:	172.29.233.254 (static)
Subnet Mask:	255.255.255.0

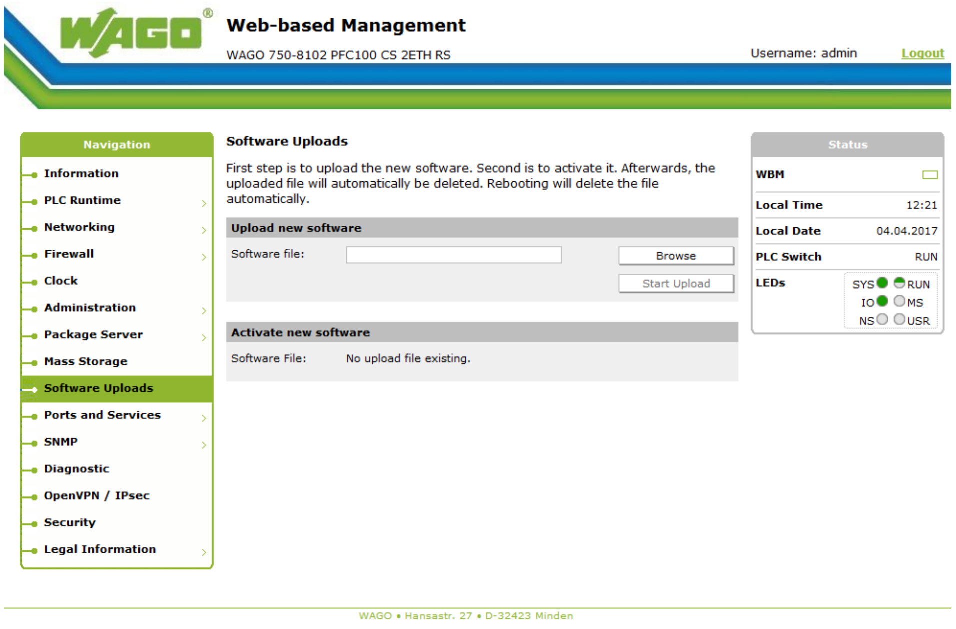
Network Details X2

State:	✓ enabled
MAC Address:	00:30:de:40:ec:d1
IP Address:	192.168.1.119 (dhcp)
Subnet Mask:	255.255.255.0

Local Time	11:21
Local Date	04.04.2017
PLC Switch	RUN

Figure 5-1: WBM Start Page before Installing the Linux Application (.ipk File)

2. Click **Software Uploads** in the menu on the left.
3. You are then prompted to authenticate. If using the default user name and password, enter “admin” in the **Username** field and “wago” in the **Password** field. Click [**Submit**] to open the Software Uploads page. You may need to confirm a security pop-up prior to that because you are using the default values for user and password.


Web-based Management
WAGO 750-8102 PFC100 CS 2ETH RS
Username: admin Logout

Navigation: Information, PLC Runtime, Networking, Firewall, Clock, Administration, Package Server, Mass Storage, Software Uploads, Ports and Services, SNMP, Diagnostic, OpenVPN / IPsec, Security, Legal Information

Software Uploads

First step is to upload the new software. Second is to activate it. Afterwards, the uploaded file will automatically be deleted. Rebooting will delete the file automatically.

Upload new software
Software file: [Browse] [Start Upload]

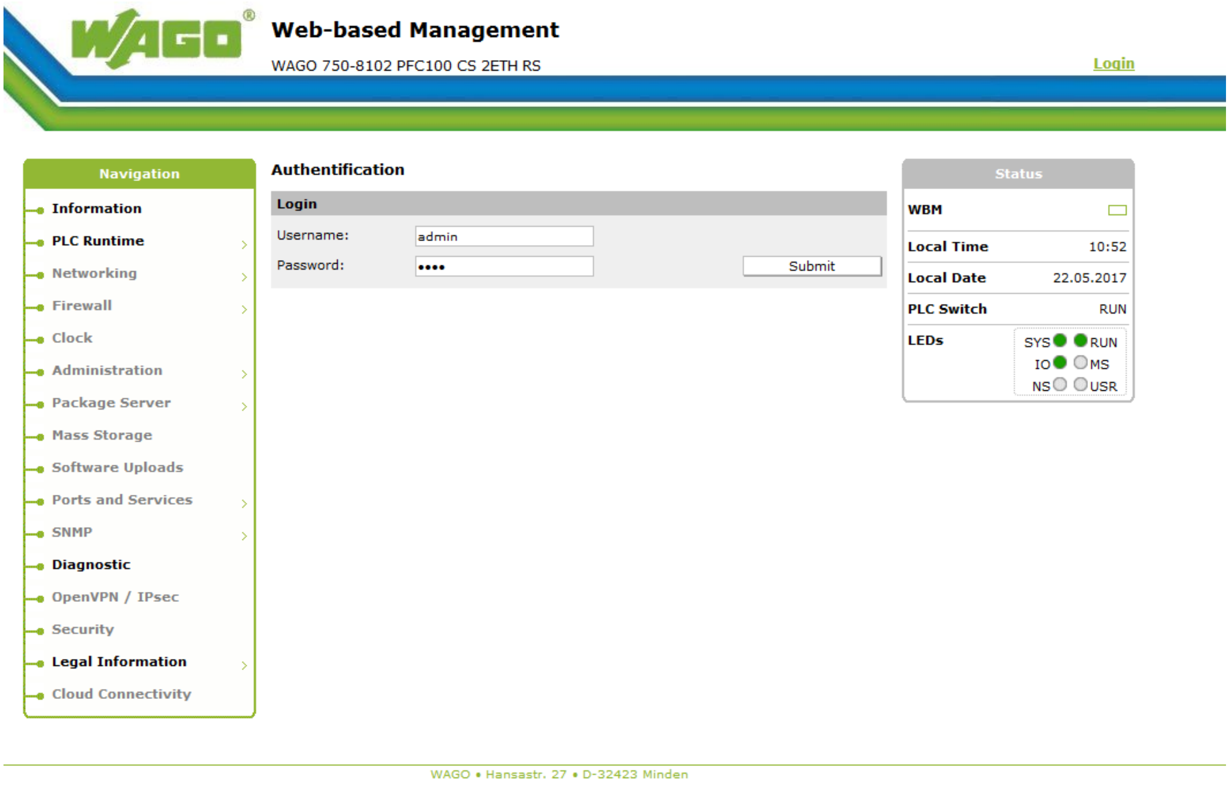
Activate new software
Software File: No upload file existing.

Status	
WBM	
Local Time	12:21
Local Date	04.04.2017
PLC Switch	RUN
LEDs	SYS RUN IO MS NS USR

WAGO • Hansastr. 27 • D-32423 Minden

Figure 5-2: Software Uploads WBM Page

4. Click [**Browse**] to open a File Upload window and navigate to the *Cloud-Connectivity_0.1.0_arm.ipk* file. Then click the file to select and then click [**Open**].
5. Then click [**Start Upload**]. The file is uploaded to the PFC.
6. Select the “Activate” **action** in the “Activate new software” group and then click [**Submit**]. Installation of the Linux Application (.ipk file) is started.
7. Once the installation is complete, reload the window in the Web browser. The WBM page shown in Figure 5-3 should be displayed. The **Cloud Connectivity** menu item has now been added. Please find a description of the Cloud Connectivity configuration for the cloud service in Section 4.


Web-based Management
WAGO 750-8102 PFC100 CS 2ETH RS
Login

Navigation: Information, PLC Runtime, Networking, Firewall, Clock, Administration, Package Server, Mass Storage, Software Uploads, Ports and Services, SNMP, Diagnostic, OpenVPN / IPsec, Security, Legal Information, Cloud Connectivity

Authentification

Login
Username: admin
Password: •••• [Submit]

Status	
WBM	
Local Time	10:52
Local Date	22.05.2017
PLC Switch	RUN
LEDs	SYS RUN IO MS NS USR

WAGO • Hansastr. 27 • D-32423 Minden

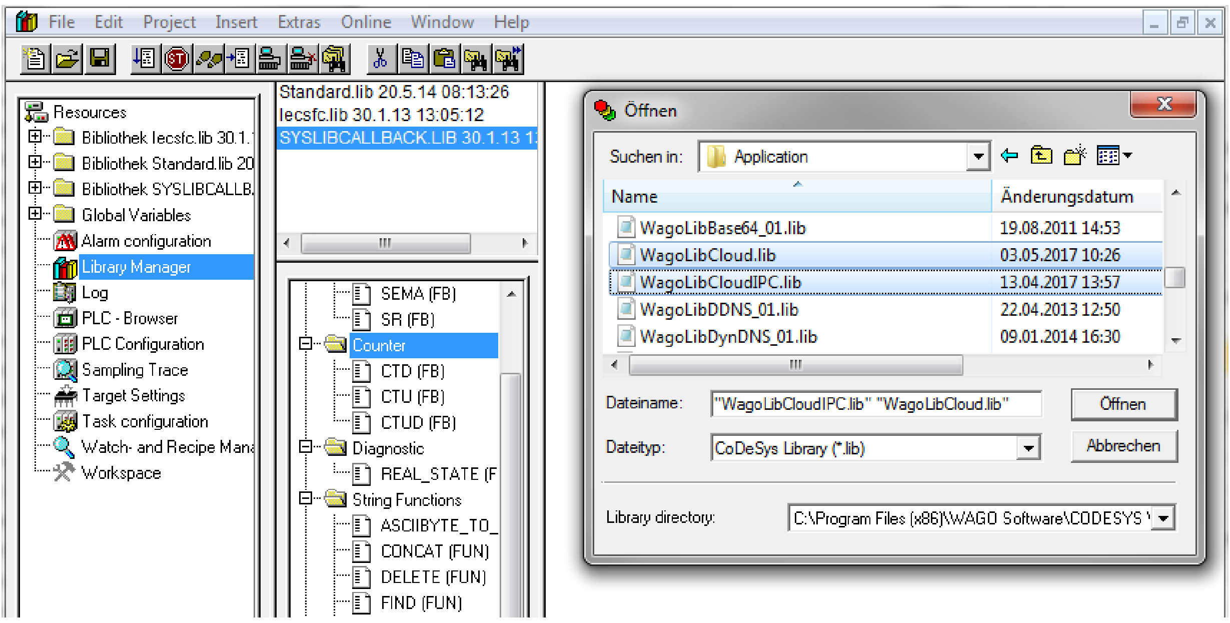
Figure 5-3: WBM Start Page after Installing the Linux Application (.ipk File)

5.1.2 IEC Libraries

5.1.2.1 Codesys 2.3

For Codesys 2.3 the libraries “WagoLibCloud.lib” and “WagoLibCloud_Internal.lib” need to be referenced.
Copy the libraries into the following path of Codesys2.3-Software:
WAGO Software \ CODESYS V2.3 \ Targets \ WAGO \ Libraries \ Application

Add the libraries using:
Resources > Library Manager > [right-click] > Additional Library…



Standard.lib 20.5.14 08:13:26
Iecsfc.lib 30.1.13 13:05:12
SYSLIBCALLBACK.LIB 30.1.13 1

Name	Änderungsdatum
WagoLibBase64_01.lib	19.08.2011 14:53
WagoLibCloud.lib	03.05.2017 10:26
WagoLibCloudIPC.lib	13.04.2017 13:57
WagoLibDDNS_01.lib	22.04.2013 12:50
WagoLibDynDNS_01.lib	09.01.2014 16:30

Dateiname: "WagoLibCloudIPC.lib" "WagoLibCloud.lib"
Dateityp: CoDeSys Library (*.lib)
Library directory: C:\Program Files (x86)\WAGO Software\CODESYS \

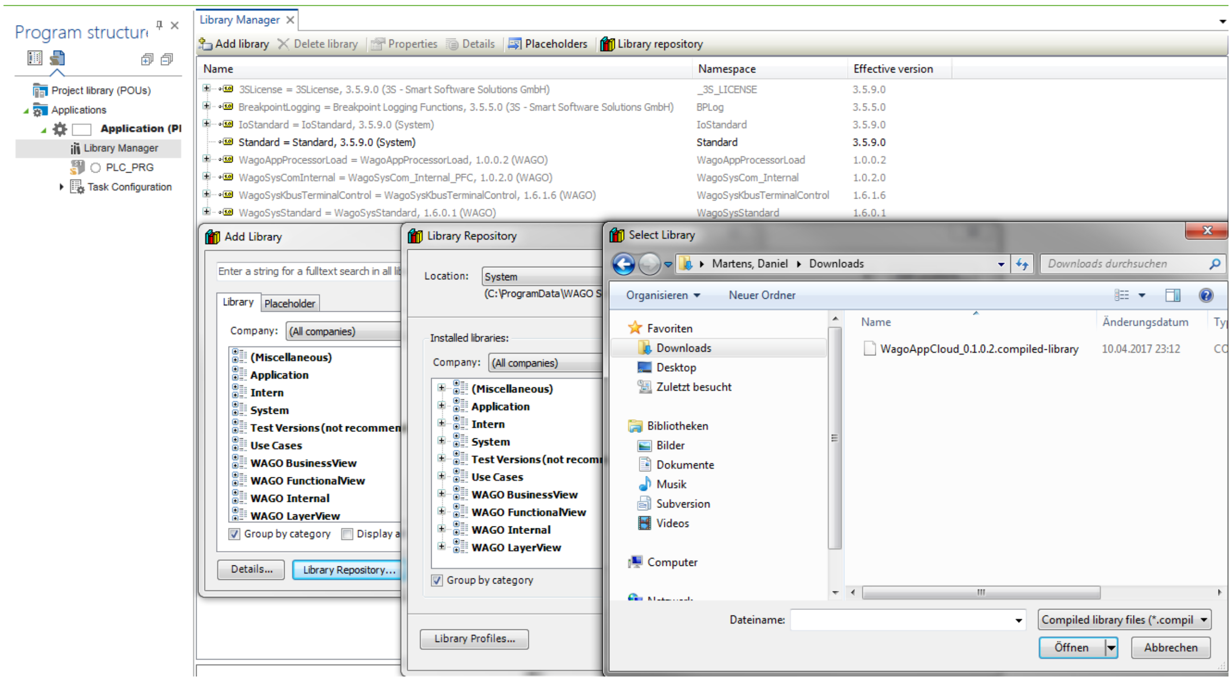
Figure 5-4: Add libraries in Codesys 2.3

5.1.2.2 e!Cockpit

For e!COCKPIT the library “WagoAppCloud_x.x.x.x.compiled-library” needs to be installed and referenced.

Install the library using

LibraryManager > Add Library > [Advanced...] > [Library Repository...] > [Install...]



Name	Namespace	Effective version
3SLicense = 3SLicense, 3.5.9.0 (3S - Smart Software Solutions GmbH)	_3S_LICENSE	3.5.9.0
BreakpointLogging = Breakpoint Logging Functions, 3.5.5.0 (3S - Smart Software Solutions GmbH)	BPLog	3.5.5.0
IoStandard = IoStandard, 3.5.9.0 (System)	IoStandard	3.5.9.0
Standard = Standard, 3.5.9.0 (System)	Standard	3.5.9.0
WagoAppProcessorLoad = WagoAppProcessorLoad, 1.0.0.2 (WAGO)	WagoAppProcessorLoad	1.0.0.2
WagoSysComInternal = WagoSysCom_Internal_PFC, 1.0.2.0 (WAGO)	WagoSysCom_Internal	1.0.2.0
WagoSysKbusTerminalControl = WagoSysKbusTerminalControl, 1.6.1.6 (WAGO)	WagoSysKbusTerminalControl	1.6.1.6
WagoSysStandard = WagoSysStandard, 1.6.0.1 (WAGO)	WagoSysStandard	1.6.0.1

Name	Änderungsdatum
WagoAppCloud_0.1.0.2.compiled-library	10.04.2017 23:12

Figure 5-5: Install the library in e!Cockpit

Note

Important note!
After an Online Change there might be changes concerning the data on certain addresses. Please regard this in case of using pointers on addresses.

5.2 SCADA System Ignition (Sparkplug)

5.2.1 Architecture

The Sparkplug specification is used for communication with the SCADA system Ignition. The Sparkplug specification is a detailed specification based on the MQTT protocol.

The following architecture picture is shown in the Sparkplug specification.

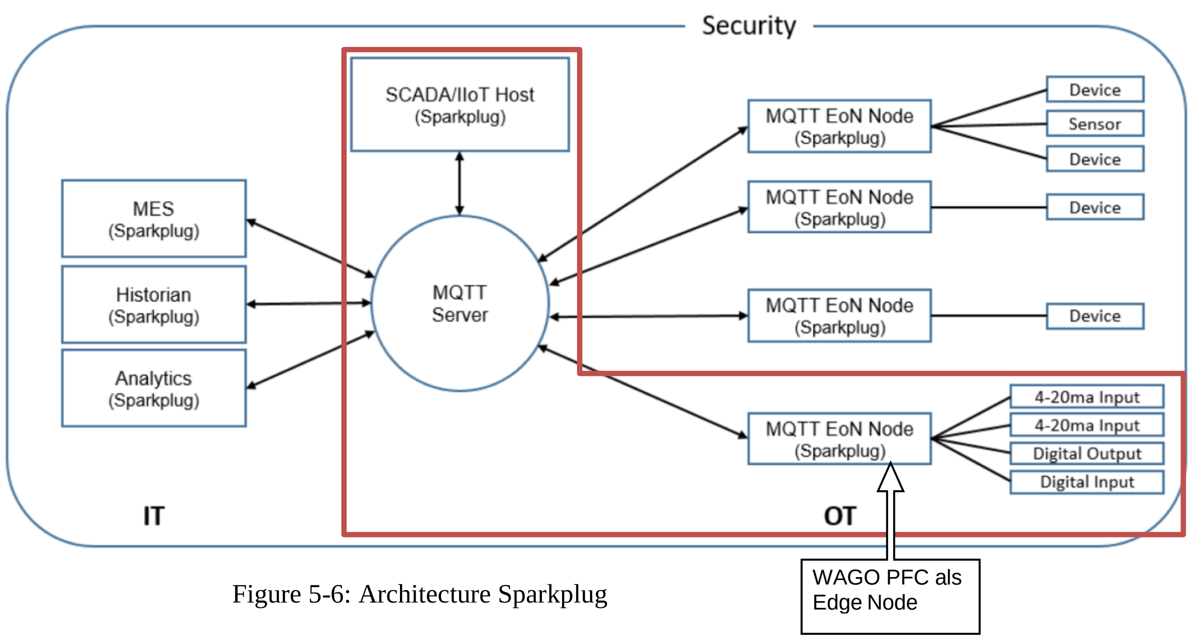


Figure 5-6: Architecture Sparkplug

The PFC is an MQTT Edge of Network Node (MQTT Edge Node) in the architecture of the Sparkplug specification. The PFC takes the measured values from the field and after configuring the data points in the PLC application, the measured values are forwarded in the format required by Sparkplug.

The requirement of Sparkplug to the MQTT Broker (MQTT Server in architecture picture) is that it is compatible with the MQTT specification version 3.1.1. It is possible to use a broker MQTT Distributor integrated in Ignition. The Broker MQTT Distributor is an add-on module in Ignition which can be installed later. The use of third-party MQTT broker is also possible.

SCADA/IIoT is the SCADA application. The SCADA application also communicates via the Sparkplug protocol. In Ignition the additional module MQTT Engine must be installed and the appropriate configuration must be entered by the MQTT Broker.

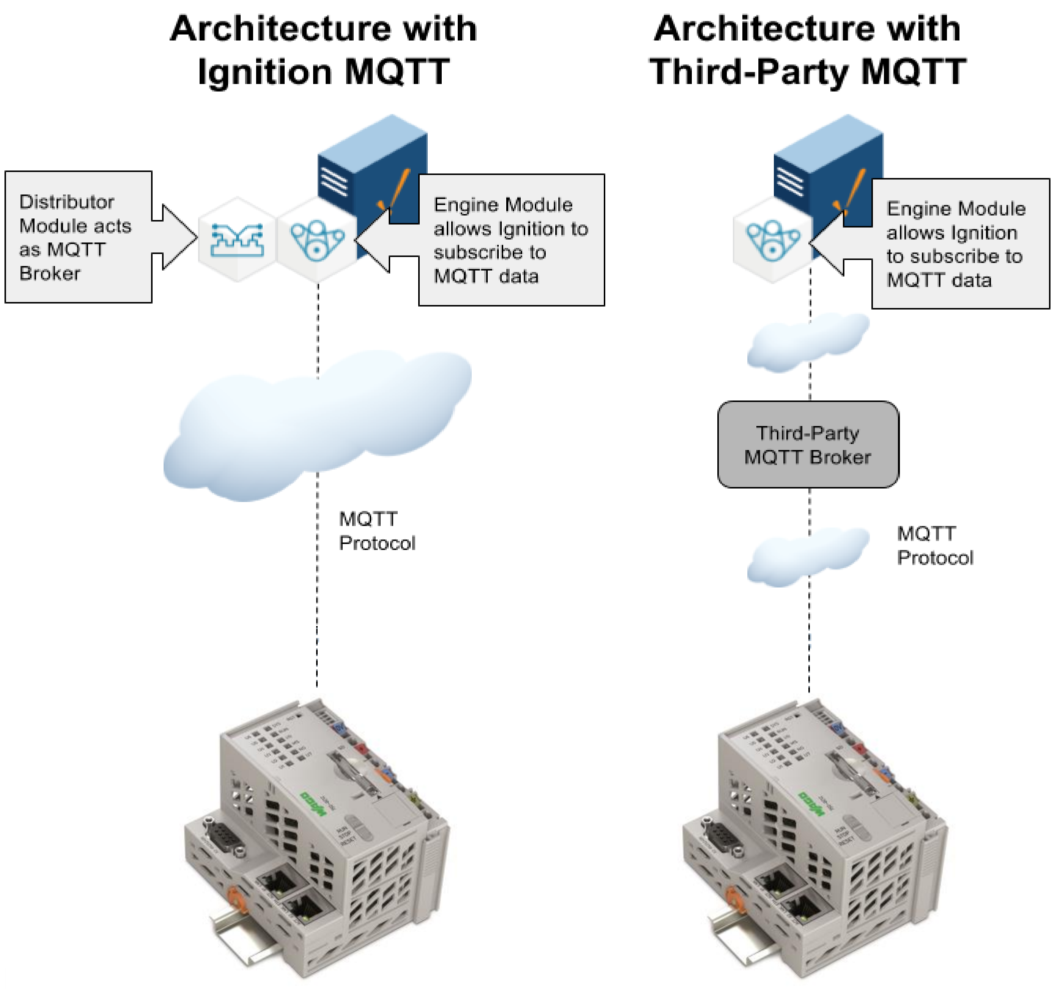

Architecture with Ignition MQTT
Distributor Module acts as MQTT Broker
Engine Module allows Ignition to subscribe to MQTT data
MQTT Protocol
Architecture with Third-Party MQTT
Engine Module allows Ignition to subscribe to MQTT data
Third-Party MQTT Broker
MQTT Protocol

Figure 5-7: MQTT Broker with Ignition

5.2.2 Configuration Cloud Connectivity with Ignition

1. Launch the PFC's WBM in a Web browser.
2. Click the **Cloud Connectivity** menu item.
3. Enter the **Username** and **Password** to authenticate on the PFC. Click [**Submit**]. A message box may appear that prompts you to change the password for security reasons. The Cloud Connectivity configuration page opens.
4. Activate the Cloud Connectivity by **Service enabled**.
5. In the "Settings" section, select the "MQTT AnyCloud" entry in the **Cloud Platform** selection box.
6. A Pop-Up window will appear which indicates the following points
   - To use the Sparkplug data protocol, a Sparkplug license required on the PFC. A test license of 30 days on the PFC is available once.
7. The **Group ID** field can be used to group individual MQTT Edge nodes (PFC) together. A Group ID must be entered.
8. Enter a name for the PFC in the **Edge node ID** field. The name of the Edge node ID is freely selectable, but should not contain any special characters and should be unique for the MQTT Broker.
9. The further configuration depends on the used MQTT Broker and its configuration.
10. Click the **[Submit]** button to write the configuration.
11. The PFC must be restarted to apply the settings.

5.2.3 Configuration in Ignition

This documentation assumes that Ignition has already been installed.
After successful installation, the following steps must be carried out:

- Install addition modules in Ignition
  - MQTT Engine
  - MQTT Distributor (optional: Installation only necessary it the internal MQTT Broker of Ignition, the MQTT Distributor, is to be used.)
- Configure the module MQTT Engine according to the settings of the used MQTT Broker
- Activate writing from the SCADA application to the PFC in the MQTT Engine module.

5.2.4 PLC Application with Ignition

To create the PLC application, it is heplful to use the following example project as a guideline:

- *WagoAppCloud_FbCommandListener_Sparkplug.ecp*

The structure of the configuration in the PLC program for the Sparkplug data protocol is very similar to that of the WAGO protocol. (see chapter 3)

The configuration of the structure and the parameters in the PLC program is mapped into the structure of Ignition.

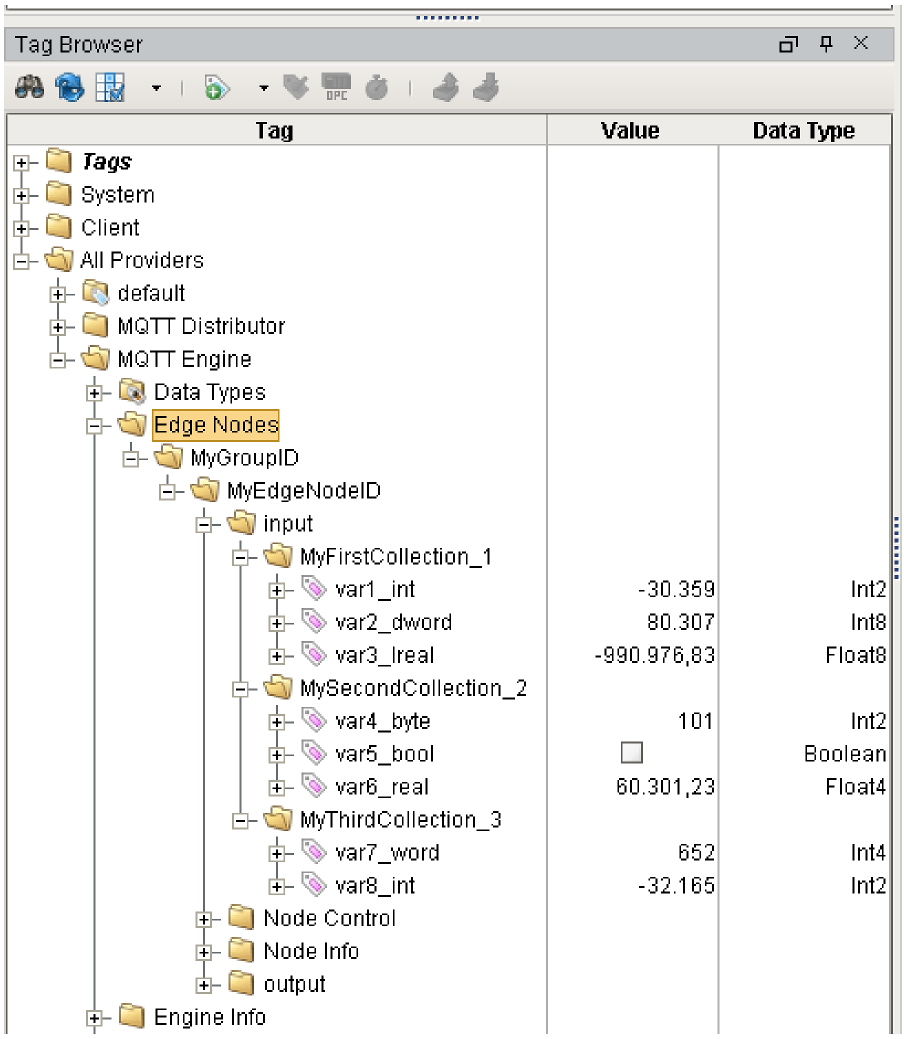
Publish data to Ignition

The directory *MyGroupID/MyEdgeNodeID/input* contains all configured collections with the corresponding variables from the PLC application.

The name of the collection in Ignition consists of the parameters *typCollection.sName* and *typCollection.dwCollectionId* from the PLC program.

Example:

- In PLC application
  - *typCollection.sName = 'MyFirstCollection'*
  - *typCollection.dwCollectionId = 1*
- In Igniton: *MyGroupID/MyEdgeNodeID/input/**MyFirstCollection_1***



Tag Browser

Tag	Value	Data Type
Tags		
System		
Client		
All Providers		
default		
MQTT Distributor		
MQTT Engine		
Data Types		
Edge Nodes		
MyGroupID		
MyEdgeNodeID		
input		
MyFirstCollection_1		
var1_int	-30.359	Int2
var2_dword	80.307	Int8
var3_lreal	-990.976,83	Float8
MySecondCollection_2		
var4_byte	101	Int2
var5_bool	☐	Boolean
var6_real	60.301,23	Float4
MyThirdCollection_3		
var7_word	652	Int4
var8_int	-32.165	Int2
Node Control		
Node Info		
output		
Engine Info		

Figure 5-8: Collection in Ignition Designer

The following table shows the mapping of the parameters from *typVariableDescription* in the PLC application to the properties in Ignition.

Configuration in PLC application	**Property in Ignition**	**Description**
typVariableDescription.sTag	Name	Name of the variable
typVariableDescription.sUnit	EngUnit	Unit of the variable
typVariableDescription.eValueType	Datatype	Data type of the variable
	‘Access Rights‘ = Read Only	Access rights in Ignition

The sent data is only sent with read rights.

Receive commands from Ignition

The directory *MyGroupID/MyEdgeNodeID/output* contains all commands from the PLC application.

The designation of the commands in Ignition consists of the parameters *typCommandDescription.sName* and *typCommandDescription.bCommandId* from the PLC application.

Example:

- In PLC application
    - *typCommandDescription.sName = ‘MyFirstCommand‘*
    - *typCommandDescription.bCommandId = 1*
- In Ignition: *MyGroupID/MyEdgeNodeID/output/**MyFirstCommand_1***

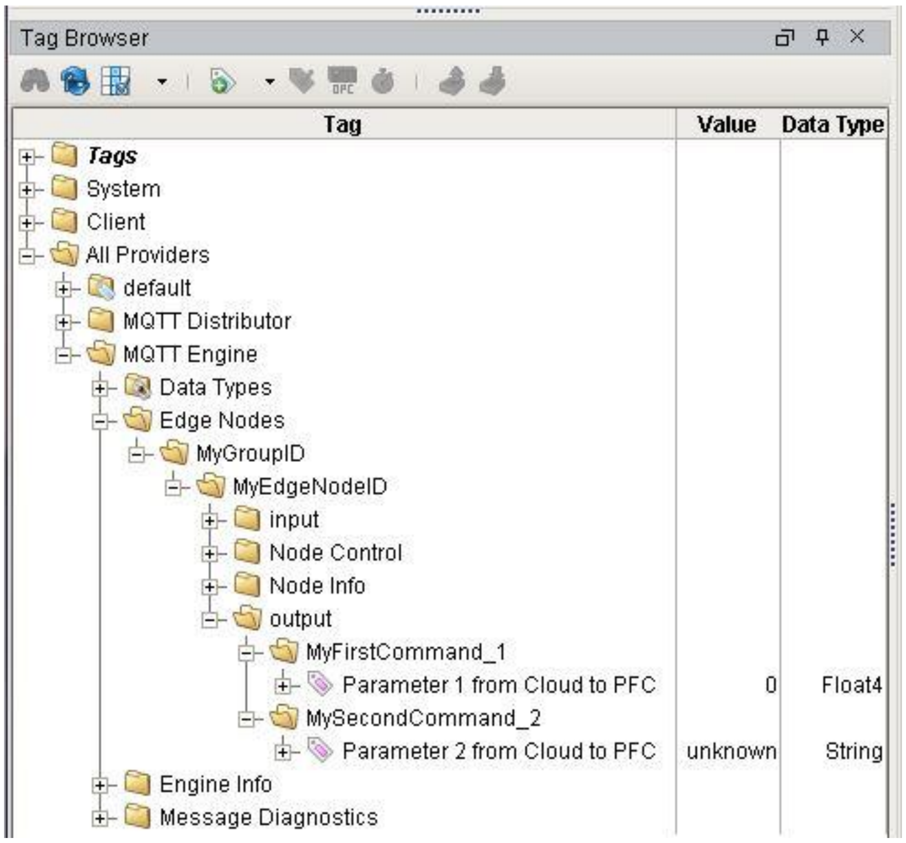


Figure 5-9: Commands in Ignition Designer

The following table shows the mapping of the parameters from *typCommandParmeterDescription* in the PLC program to the properties in Ignition.

Configuration in PLC application	**Property in Ignition**	**Beschreibung**
typCommandParameterDescription. sParameterName	Name	Name of the variable
typCommandParameterDescription. eParamterType	Datatype	Data type of the variable
	‘Access Rights‘ = Read/Write	Access rights in Ignition

The commands are received in the PLC program in the function block FbCommandListener and can be further evaluated there.

Note that the Sparkplug specification does not support CommandResponse. Therefore the following points must be observed in the PLC application:

- No use of FbCommandResponder
- *typCommandDescription.bNumberOfResponseParameters = 0*

Examples: Policies for AWS

5.2.5 Policy without curtailment

```
{
  "Version": "2019-02-25",
  "Statement": [
    {
      "Action": [
        "iot:Publish",
        "iot:Subscribe",
        "iot:Connect",
        "iot:Receive"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    }
  ]
}
```

5.2.6 Possible Policy for WAGO Protocol v1.0

This example has been executed with the following settings:

- Account 1234-5678-9123 in the region New Virginia (us-east-1)
- WAGO Protocol Version 1.0
- Client ID in WBM: ‘PFC200’

```
{
 "Version": "2019-02-26",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish",
    "iot:Receive"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789123:topic/PFC200/CloudHello",
    "arn:aws:iot:us-east-1:123456789123:topic/PFC200/Commands",
    "arn:aws:iot:us-east-1:123456789123:topic/PFC200/DeviceHello",
    "arn:aws:iot:us-east-1:123456789123:topic/PFC200/DeviceInfo",
    "arn:aws:iot:us-east-1:123456789123:topic/PFC200/DeviceState",
    "arn:aws:iot:us-east-1:123456789123:topic/PFC200/TagConfiguration",
    "arn:aws:iot:us-east-1:123456789123:topic/PFC200/TagValues",
    "arn:aws:iot:us-east-1:123456789123:topic/PFC200/EventTagValues",
    "arn:aws:iot:us-east-1:123456789123:topic/PFC200/CommandRegistration",
    "arn:aws:iot:us-east-1:123456789123:topic/PFC200/CommandResponse"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Subscribe"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789123:topicfilter/PFC200/CloudHello",
    "arn:aws:iot:us-east-1:123456789123:topicfilter/PFC200/Commands"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789123:client/PFC200"
   ]
  }
 ]
}
```

5.2.7 Possible Policy for WAGO Protocol v1.2

This example has been executed with the following settings:

- Account 1234-5678-9123 in the region New Virginia (us-east-1)
- WAGO Protocol Version 1.2
- Client ID in WBM: ‘PFC200’

```
{
 "Version": "2019-02-26",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish",
    "iot:Receive"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789123:topic/PFC200/CloudHello",
    "arn:aws:iot:us-east-1:123456789123:topic/PFC200/DeviceHello",
    "arn:aws:iot:us-east-1:123456789123:topic/1.2/PFC200/Commands",
    "arn:aws:iot:us-east-1:123456789123:topic/1.2/PFC200/DeviceInfo",
    "arn:aws:iot:us-east-1:123456789123:topic/1.2/PFC200/DeviceState",
    "arn:aws:iot:us-east-1:123456789123:topic/1.2/PFC200/TagConfiguration",
    "arn:aws:iot:us-east-1:123456789123:topic/1.2/PFC200/TagValues",
    "arn:aws:iot:us-east-1:123456789123:topic/1.2/PFC200/CommandRegistration",
    "arn:aws:iot:us-east-1:123456789123:topic/1.2/PFC200/CommandResponse"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Subscribe"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789123:topicfilter/PFC200/CloudHello",
    "arn:aws:iot:us-east-1:123456789123:topicfilter/1.2/PFC200/Commands"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789123:client/PFC200"
   ]
  }
 ]
}
```

WAGO Kontakttechnik GmbH & Co. KG

Postbox 2880 •	D-32385 Minden
Hansastraße 27 •	D-32423 Minden
Phone:	+49 (0) 571/8 87 – 0
Fax:	+49 (0) 571/8 87 – 1 69
E-Mail:	info@wago.com
Internet:	http://www.wago.com