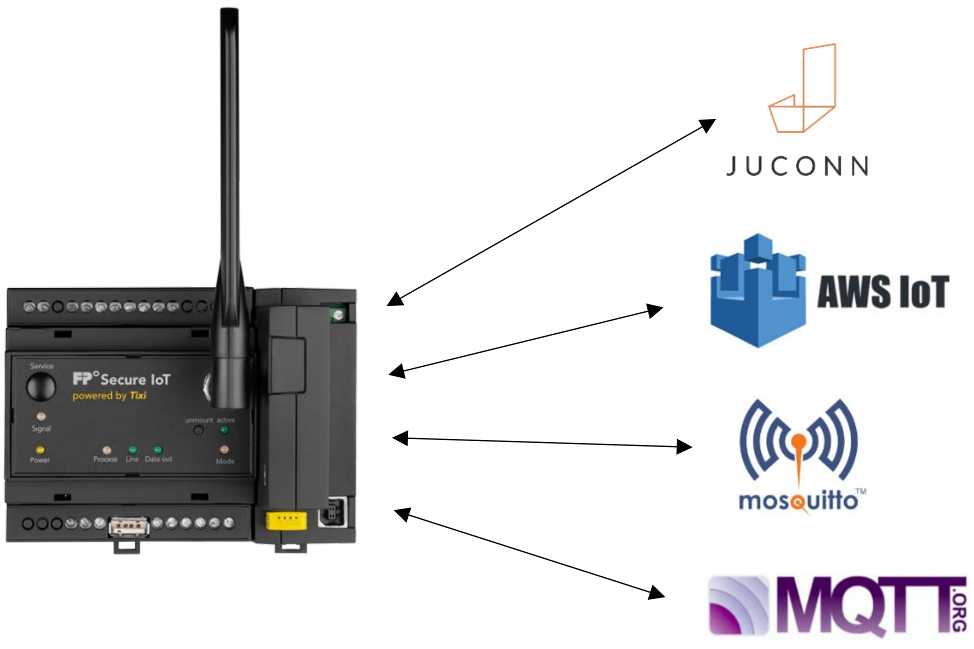
FP Juconn MQTT Client for FP Gateways Manual



Version: 1.3.1

Publication date: 23/02/2021

Table of contents

1 Introduction

The FP-Juconn MQTT client is a universal MQTT client that can be used for both the Juconn platform and AWS equally. However, in principle, the MQTT client can also be connected to any other MQTT broker that supports the current MQTT standard “MQTT 3.1.1”.

Rollout, initialisation and normal operation of the gateway
After setting up the gateway for the first time with an initial TiXML configuration, the FP gateway must first run through a rollout process (“gatewayRollout”) with the portal. During the rollout process, the gateway receives its unique gateway ID and customer ID from the broker.
These two IDs are later used as part of all sending topics.

If the gateway ID and the customer ID are already predefined in the CloudConnector database, the rollout can also be skipped.

After the rollout and during each subsequent gateway restart, an initialisation routine (“gatewayInit”) must run, in which the gateway transmits its data model to the broker.

Once initialisation is complete, the gateway switches to normal mode (running status) and sends data to the broker. Furthermore, the gateway can receive and evaluate commands via the feedback channel.

Sending methods and sending intervals
As soon as an MQTT connection has been established (running status), the client sends the process data or the log data to the broker in the set cycle. The option to set up a hysteresis function is also available, as well as sending event-based data. Furthermore, log data can be sent (e.g. if communication to the broker was interrupted for an extended period). If necessary, an MQTT caching mechanism can also be used for this.

The gateway’s data can be sent to the portal in the following way:

1. Process data (process/…)
    a. cyclically
    b. Event-controlled via “CloudSendRealtimeData” sending command
    c. Hysteresis function
2. Log data
    a. Cyclically
    b. Event-controlled

Feedback channel to control the gateway via the broker
There are several feedback channels that the broker can use to send TiXML commands to the client. The results of the commands are then sent back to the broker via another topic (see Table 1-1).

Data format
The data can be sent either in JSON format or in TiXML format. The data format is configured in the CloudConn database (**`CloudConnDataFormat`**, see section 7).
In Version 1, only the TiXML data format is supported.

Topics
Most topics comprise a **`fixed part`** and a **`variable part`**. Example: Topic to send all data for a gateway (publish):
`$customer_id/fp/$gateway_id/data`
The **`$customer_id`** and the **`$gateway_id`** are the result of the rollout process and identify a gateway uniquely within a broker.

From the gateway’s point of view, the following topics exist:

Topic	Direction	Meaning
fp/gatewayRollout	publish	Start rollout process
fp/gatewayRolloutResult/$serial_number	subscribe	Receive the rollout result from the broker. Each gateway has its own response topic (data security)
$customer_id/fp/$gateway_id/gatewayInit	Publish	Start initialisation process
$customer_id/fp/$gateway_id/gatewayInitResult	subscribe	Receive the initialisation process result from the broker
$customer_id/fp/$gateway_id/data	publish	Send all data Standard sending topic for real-time data
$customer_id/fp/$gateway_id/config	subscribe	The gateway receives the SetConfig command
$customer_id/fp/$gateway_id/configResult	publish	The gateway sends the result of the SetConfig command
$customer_id/fp/$gateway_id/cmd	subscribe	The gateway receives arbitrary TiXML commands
$customer_id/fp/$gateway_id/cmdResult	publish	The gateway sends the result of the commands that were executed
$customer_id/fp/$gateway_id/state	subscribe	The gateway receives the command to send general gateway information
$customer_id/fp/$gateway_id/stateResult	publish	The gateway sends general gateway information to the broker

Table 1-1: Overview of topics used

Connection check
An MQTT broker can be configured in the client. The client then attempts to establish a connection to this broker after the start.

Encrypted vs. unencrypted
The connection to the broker can either be established unencrypted (port 1883) or encrypted (port 8883).

1.1 Commissioning process

FP gateways are pre-set with the suitable broker URL in the factory.
If the devices are then delivered to the customer, the run through the commissioning process with the rollout (optional) and initialisation stages. They are then able to send data to the broker.

During the rollout, the gateway logs on to the broker using its product name and serial number. The response from the broker assigns the gateway a $gateway_id, a $customer_id and, if necessary, a broker URL for further communication. Furthermore, an initial configuration can also be transmitted to the gateway.

This completes the rollout process. The rollout process can also be skipped. In this case, the $gateway_id and the $customer_id re pre-configured.

During the subsequent initialisation, the FP gateway informs the broker of which sensors are connected (data model). Meta data such as the sensor type, unit, etc. is also transmitted.

After initialisation is complete, the FP gateway then sends the data agreed upon during the initialisation process (e.g. sensor data).

Process steps during commissioning (see Table 1-2):
ROLLOUT –> WAIT FOR ROLLOUT RESULT –> INITIALISIERUNG -> WAIT FOR INIT RESULT –> RUNNING

Process step	Status description
ROLLOUT (optional)	Gateway delivery state to the customer; Juconn URL predefined The gateway logs on to the broker and sends a serial number and the device name
WAIT FOR ROLLOUT RESULT (optional)	The gateway expects the first information from the broker ($customer_id, $gateway_id; optional broker URL and gateway configuration)
INITIALISIERUNG	The gateway calls the topic $customer_id/fp/$gateway_id/gatewayInit and sends its configuration to the broker
WAIT FOR INIT RESULT	The gateway waits for a response from the broker
RUNNING	The gateway sends the sensor data; it reacts to broker commands

Table 1-2: Lifecycle status overview

Process step	Topics: Gateway -> Broker	Topics: Broker -> Gateway
ROLLOUT	fp/gatewayRollout	
WAIT FOR ROLLOUT RESULT		fp/gatewayRolloutResult/$serial_number
INITIALISIERUNG	$customer_id/fp/$gateway_id/gatewayInit	
WAIT FOR INIT RESULT		$customer_id/fp/$gateway_id/gatewayInit Result
RUNNING	$customer_id/fp/$gateway_id/data	

Table 1-3: Topic assignment lifecycle

1.2 MQTT payload data format

The gateway can send the payload to the broker in JSON and TiXML format.
The data format is defined in the CloudConn database using the CloudConnDataFormat parameter.
The default setting is TiXML. The gateway must be able to process both data formats.

1.3 Gateway error messages

If the gateway receives commands from the broker and executes these commands, error messages are sometimes generated if the command contains incorrect parameters for example, or cannot be executed for other reasons.

A parameter can be used when sending the command to specify how comprehensive the response to the gateway error messages is.

There are three possible formats that can be used to transmit these errors. A short or a long error message is issued depending on the specification by the “ver” parameter. See Table 1-4

Command example:
```
    [<Get _="/Processs/" ver="vmode"/>]
```

vmode	Description	Example
n	Error message as numeric value	`<Error _="-2196" />`

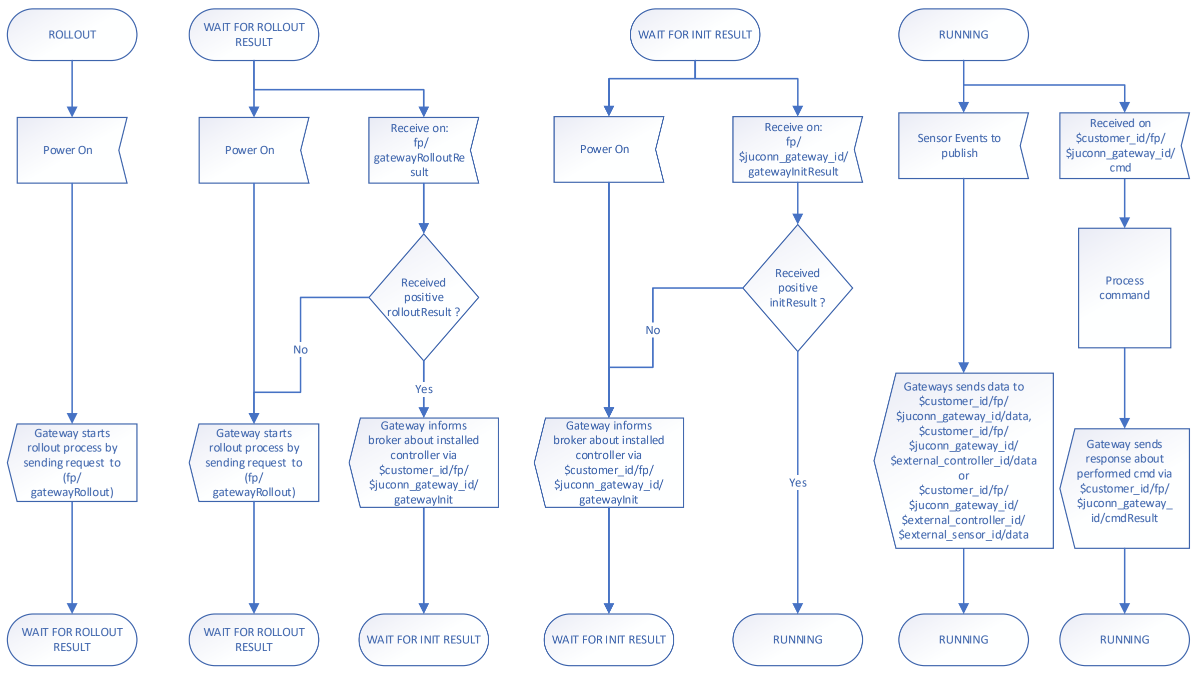
y	Error message, short description	`<Error>`
		`  <ErrNo _="-2196" />`
		`  <ErrText _="path to key not found" />`
		`  <ErrorCause>`
		`    <ErrNo _="-2196" />`
		`    <ErrText _="path to key not found" />`
		`    <Class _="TXSTCPGetSetValueCmd" />`
		`  </ErrorCause>`
		`</Error>`

v	Error message, long description	`<Error>`
		`  <ErrNo _="-2196" />`
		`  <ErrText _="path to key not found" />`
		`  <ErrorCause>`
		`    <ErrNo _="-2196" />`
		`    <ErrText _="path to key not found" />`
		`    <Line _="127" />`
		`    <Module _="SSet.cpp" />`
		`    <Class _="TXSTCPGetSetValueCmd" />`
		`  </ErrorCause>`
		`</Error>`

Table 1-4: Error return values depending on the “ver” parameter

2 The status of the gateway lifecycles

Gateway Lifecycle


ROLLOUT
Power On
Gateway starts rollout process by sending request to (fp/ gatewayRollout)
WAIT FOR ROLLOUT RESULT
WAIT FOR ROLLOUT RESULT
Power On
Gateway starts rollout process by sending request to (fp/ gatewayRollout)
WAIT FOR ROLLOUT RESULT
Receive on: fp/ gatewayRolloutResult
Received positive rolloutResult ?
No
Yes
Gateway informs broker about installed controller via $customer_id/fp/ $juconn_gateway_id/ gatewayInit
WAIT FOR INIT RESULT
WAIT FOR INIT RESULT
Power On
Gateway informs broker about installed controller via $customer_id/fp/ $juconn_gateway_id/ gatewayInit
WAIT FOR INIT RESULT
Receive on: fp/ $juconn_gateway_id/ gatewayInitResult
Received positive initResult ?
No
Yes
RUNNING
RUNNING
Sensor Events to publish
Gateways sends data to $customer_id/fp/ $juconn_gateway_id/data, $customer_id/fp/ $juconn_gateway_id/ $external_controller_id/data or $customer_id/fp/ $juconn_gateway_id/ $external_controller_id/ $external_sensor_id/data
RUNNING
Received on $customer_id/fp/ $juconn_gateway_id/ cmd
Process command
Gateway sends response about performed cmd via $customer_id/fp/ $juconn_gateway_ id/cmdResult
RUNNING

See [1]

2.1 Rollout (gatewayRollout)

2.1.1 Start rollout

The gateway sends the following data to the broker using the URL defined with CloudBaseUrl in section 7. If the `gateway_id` and `customer_id` parameters are predefined in the CloudConn database, the rollout is skipped.

Topic

```
fp/gatewayRollout
```

Body

```
{
     "request_id": "generated string",
     "prod_name": "$tixi_prodname",
     "serial_number": "$tixi_gateway_serial_number"
}
```

… or in TiXML:

```
{
     <?xml version="1.0" encoding="UTF-8" ?>
     <request_id _="generated string"/>
     <prod_name _="$tixi_prodname"/>
     <serial_number _="$external_gateway_serial_number" />
}
```

The broker checks whether the gateway serial number is known and whether the rollout process is expected for this gateway.

During this communication, the FP gateway defines a `request_id`.

```
Layout of the request_id during rollout:
Device_serial_number_x
Where x is increased with each rollout request. The use of the
serial number ensures that the request_id can be used uniquely
system-wide and only by one device.
```

The gateway's serial number (`$tixi_gateway_serial_number`) can be queried using the following path: `http://[Gateway IP]/System/Properties/SerialNo`

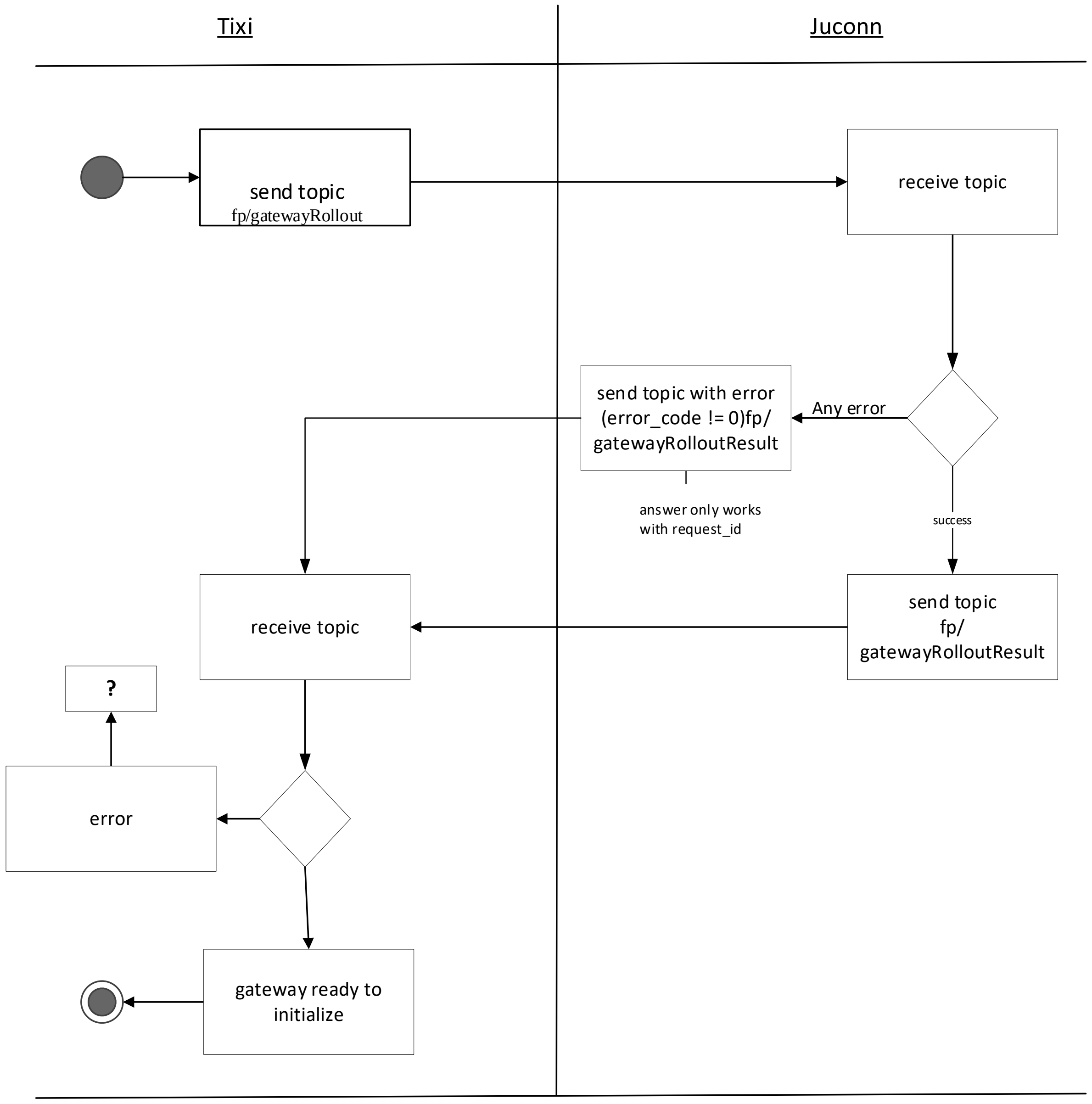
The gateway’s product name (`$tixi_prodname`) can be queried using the following path: `http://[Gateway IP]/System/Properties/Hardware/Modules/Modem0`

Example body for a WT640 wall box with the serial number `01234567`:

```
{
     <?xml version="1.0" encoding="UTF-8" ?>
     <request_id _="01234567_1"/>
     <prod_name _="WT640"/>
     <serial_number _="01234567" />
}
```

The device serial number contains 8 characters and comprises only numbers.

Rollout



Tixi
Juconn
send topic fp/gatewayRollout
receive topic
send topic with error (error_code != 0)fp/ gatewayRolloutResult
Any error
answer only works with request_id
success
send topic fp/ gatewayRolloutResult
receive topic
?
error
gateway ready to initialize

2.1.2 Wait for rollout result

The gateway waits for the response from the broker that answers the rollout request as follows:

Topic

```
fp/gatewayRolloutResult/$serial_number
```

Body

```
{
     "error_code": 0,
     "response_id": "the received request_id from fp/gatewayRollout",
     "gateway_id": "Object_ID",
     "customer_id": "Object_ID",
     //--------- Start of optional area --------
     "data_broker_details": {
        "url": "broker.juconn.io",
        "qos": 0
    },
     {
       "gateway_configuration": "gateway configuration details"
     }
     //---------- End of optional area ---------

}
```

... or in TiXML:

```
{
     <?xml version="1.0" encoding="UTF-8" ?>
     <error_code _="0" />
     <response_id _="the received request_id from fp/gatewayRollout"/>
     <gateway_id _="Object_ID" />
     <customer_id _="Object_ID" />

     <!--------- Start of optional area -------->
     <data_broker_details>
        <url _="broker.juconn.io"/>
        <qos _="0"/>
     </data_broker_details>

     <gateway_configuration>
        gateway configuration details
     </gateway_configuration>
     <!---------- End of optional area --------->
}
```

$serial_number is the gateway's real serial number (e.g. 04125378).

If the `response_id` does not match the `request_id` from the previous publish (fp/gatewayRollout), the rollout process must be executed again.
The “gateway_id” and “customer_id” credentials transmitted by the broker must be saved permanently in the gateway and are used later as a path specification in all topics.
During a factory reset or a `<Reset _="Secretfile"/>`, the credentials must be deleted again.
The Portal_Version variable is used to transmit the broker’s software version to the gateway. This value is intended to help Support in particular. For information regarding displaying the Portal_Version, see section 7.5.

The optional “gateway_configuration” can be used to transmit a configuration to the FP gateway from the broker. The customer can therefore obtain the same configuration on all of their gateways for example. The “gateway_configuration” contains one or several FP databases in JSON or TiXML format.
The “gateway_configuration” must be activated from the gateway. To do so, execute the SetConfig commands individually and enter the result into each “SupportLog” system log file.

Example of a gateway_configuration:

JSON
```
{
 "gateway_configuration": {
  "SetConfig": [
     {
        "-_": "USER",
        "-ver": "y",
        "AccRights": {
           "Groups": {
              "Admin": {
                 "CardLogin": {"-AccLevel": 1},
                 "EthernetLogin": {"-AccLevel": 1},
                 "LocalLogin": {"-AccLevel": 1},
                 "RemoteLogin": {"-AccLevel": 1}
              },
              "Webserver": {"WebServer": {"-AccLevel": 20}
              }
           },
           "User": {"-_": "Plain","ADMIN":{"-Plain":"","-Group":"Admin"},
              "Def_CardLogin": {"-Plain": "", "-Group": "Admin"},
              "Def_EthernetLogin": {"-Plain": "", "-Group": "Admin"},
              "Def_LocalLogin": {"-Plain": "", "-Group": "Admin"},
              "Def_RemoteLogin": {"-Plain": "", "-Group": "Admin" }
           }
        }
     },
     {
        "-_": "ISP",
        "-ver": "y",
        "WLAN_AP": {"AllowedConnections": {"-_": 1},
           "DisableAfter": {"-_": 0},
           "EnableOnStartup": {"-_": 1}
        }
     }
  ]
 }
}
```

… or in TiXML:
```
<gateway_configuration>

   <!-- Einer oder mehrere SetConfig-Befehle -->
    <SetConfig _="USER" ver="y">
      <AccRights>
         <Groups>
           <Admin>
              <LocalLogin AccLevel="1"/>
              <CardLogin AccLevel="1"/>
              <RemoteLogin AccLevel="1"/>
              <EthernetLogin AccLevel="1"/>
            </Admin>

            <Webserver>
              <WebServer AccLevel="20"/>
            </Webserver>
          </Groups>

          <User _="Plain">
            <Def_LocalLogin Plain="" Group="Admin"/>
            <Def_CardLogin Plain="" Group="Admin"/>
            <Def_RemoteLogin Plain="" Group="Admin"/>
            <Def_EthernetLogin Plain="" Group="Admin"/>
            <ADMIN Plain="" Group="Admin"/>
          </User>
       </AccRights>
    </SetConfig>

    <SetConfig _="ISP" ver="y">
      <WLAN_AP>
        <EnableOnStartup _="1"/>
        <AllowedConnections _="1" />
        <DisableAfter _="0" />
      </WLAN_AP>
    </SetConfig>

</gateway_configuration>
```

In the next step, “Gateway initialisation”, the sensors and actuators connected to the gateway are transmitted to the broker.

2.2 Gateway initialisation (gatewayInit)

2.2.1 Start gatewayInit

During the “Gateway initialisation”, the gateway sends the definition of all data points to the portal, which are to be processed there later. This is used to align the gateway’s data model with the portal’s data model. Gateway initialisation is performed each time the gateway logs in to the portal so that the gateway data model always matches the portal's data model.

Topic

```
$customer_id/fp/$gateway_id/gatewayInit
```

Body

Basic body layout:

```
{
     "request_id": "generated string",
     //--------- Start of optional area --------
     "virtual_devices": "virtual_devices_flag",
     "gateway_details": "TBD: gateway details",
     //--------- Start of optional area --------
     "controllers": [
       {
        controller 1 details
       },
       {
        controller 2 details
       },
       {
        controller n details
       }
    ]
}
```

Definition of `virtual_devices`

The optional “`virtual_devices`” flag defines whether data points for virtual devices (controllers) are to be summarised in the cloud. The flag is defined in the CloudConn database using the “`virtual_devices`” tag. Possible values are:
0: Cloud should not create any virtual devices (default value; behaviour as up to now)
1: Cloud should create virtual devices (new behaviour)

If the “virtual_devices” CloudConn flag is set to 1, the cloud business logic should only create virtual devices (controllers). Data points are assigned to the virtual devices using the “`meta_virtual_devices`” tag that must be defined for each data point in the realtime sections on the CloudConn database.
If the “virtual_devices” CloudConn flag is set to 1, all data points for all realtime sections must be assigned virtual devices using the “`meta_virtual_device`” tag. The virtual device’s name can be freely defined but should only use ASCII characters and must not contain any spaces.

Controller details

The “controller details”contain all of the gateway’s data points that are then to be sent to the broker cyclically or via an event at a later time (data model = device inventory). Meta data to classify the data points precisely is also transmitted.

A Juconn controller is a device in the gateway world.

Definition of controller name

1. **Variable access path > 1st level**
   Controller name = path name up to the last hierarchy but one
   $Controller_ name_x designates the content of the corresponding variables
   $Sensor_id_x designates the content of the corresponding variables
   Slashes are replaced by underscores

Examples (the sensor ID is **bold** in the table):

FP variable path (Get..)	Controller name	Comment
`/GSM/`**`State`**	`GSM`	GSM information
`/Process/PV/`**`Var_PV`**	`Process_PV`	ProcessVars
`/Process/CloudConn/`**`ConnectionState`**	`Process_CloudConn`	CloudConn status
`/Process/VPN/`**`ConnectionState`**	`Process_VPN`	VPN Status
`/Process/Bus1/Device0/`**`Var0`**	`Process_Bus1_Device0`	Bus devices
`/Process/C540/Q/`**`P0`**	`Process_C540_Q`	S1 module, bit access
`/Process/C540/QB/`**`P0`**	`Process_C540_QB`	S1 module, byte access
`/Process/C540/QW/`**`P0`**	`Process_C540_QW`	S1 module, Word access
`/Process/C540/QD/`**`P0`**	`Process_C540_QD`	S1 module, DWord access
`/Process/C03e/Counter/`**`P0`**	`Process_C03e_Counter`	Expansion module
`/Process/MB/IO/I/`**`P0`**	`Process_MB_IO_I`	Digital input
`/Process/MB/IO/Q/`**`P0`**	`Process_MB_IO_Q`	Digital output

Table 2-1: Variable access path > 1st level

2. Variable access path = 1st level
   Controller name = Get
   Slashes are replaced by underscores

Examples (the variable name is **bold** in the table):

FP variable path (Get..)	Controller name	Comment
`/SerialNo`	`Get`	Serial number
`/GprsLinkState`	`Get`	GPRS link status
`/FreeFileSize`	`Get`	Free flash memory

Table 2-2: Variable access path = 1st level

Definition of controller meta data

The following meta data is optional. How this data is stored in the TiXML configuration must still be specified.

```
"custom_fields": null,
"timeout": 360,
```

Definition of sensor data

`external_sensor_id:` unique within the controller

`name:` Name of the sensor; unique sensor name; the content is designated with $Varname_x below
-> results from the data point's XML name in the CloudConn database

```
<Realtime1>
   <$Varname_1 _="/Process/PV/MyProcessVar" />
</Realtime1>
```

In the example above, $`Varname_1` is the name of the sensor.

Defining meta data for sensors

The sensor's meta data is defined in the realtime branches (a maximum of 5 branches can be defined) in the CloudConnector database for each data point. The data is defined there as additional tags in the variable's access path:

```
<Realtime1>
   <$Varname_1 _="/Process/PV/MyProcessVar1"
       meta_type="humidity" meta_unit="%" meta_setpoint="30"
       meta_virtual="true" />
   <$Varname_2 _="/Process/MBus/Device0/Var0"
       meta_type="light" meta_unit="lm" meta_receive_signal="20" />
</Realtime1>
```

All tags defined in the CloudConn database must be translated to the Juconn name by removing the text "`meta_`" from the tag name. Therefore, `meta_type="temperature"` becomes `type="temperature"`. These Juconn tags are then transmitted to the portal as sensor meta data. Table 2-3 lists all possible meta data.

New: Virtual devices

Any number of data points can be assigned to virtual devices in the cloud. This grouping enables the user to summarise data from various sources and therefore to display it clearly in the cloud dashboard.
To do so, the "`meta_virtual_device`" XML tag is added to the meta data in the realtime sections, which is used to define the name of the virtual devices and therefore to group the data. The parameter is transmitted during the `gatewayInit` phase via the `virtual_device` meta data parameter.

The group names defined by the "`meta_virtual_device`" XML tag are used in the cloud as the names for the virtual devices (controller name).

Example:

```
<Realtime1>

  <Varname_1  _="/Process/PV/MyProcessVar1" meta_type="humidity"
                meta_unit="%" meta_setpoint="30"
                meta_virtual_device="Heating_1"/>

  <Varname_2  _="/Process/MBus/Device0/Var0" meta_type="light"
                meta_unit="lm" meta_receive_signal="20"
                meta_virtual_device="Heating_1" />
  <Varname_3  _="/Process/PV/MyProcessVar2" meta_type="temperature"
                meta_unit="K" meta_setpoint="5"
                meta_virtual_device="Heating_2"/>
  <Varname_4  _="/Process/MBus/Device1/Var0" meta_type="power"
                meta_unit="W" meta_receive_signal="10"
                meta_virtual_device="Heating_2"/>
  <Varname_5  _="/Process/Modbus/Device0/Var0" meta_type="voltage"
                meta_unit="V" meta_virtual_device="Heating_2" />

</Realtime1>
```

In the example, the `Varname_1` and `Varname_2` data points are assigned to the "**`Heating_1`**" virtual device. The `Varname_3`, `Varname_4` and `Varname_5` data points are assigned to the "**`Heating_2`**" virtual device.

Excerpt from the `gatewayInit` payload (only sensors are shown here for simplicity) for the two ProcessVars (PV) from the example above:

```
    <sensors>
       <$sensor_id_1>
         <name _="Varname_1"/>
         <type _="humidity"/>
         <unit _="%"/>
```

```
      <set_point _="30" />
      <virtual_device _="Heating_1"/>
   </$sensor_id_1>
   <$sensor_id_2>
      <name _="Varname_2"/>
      <type _="temperature"/>
      <unit _="K"/>
      <set_point _="5" />
      <virtual_device _="Heating_2"/>
   </$sensor_id_2>
</sensors>
```

The number of data points assigned to a virtual device is unlimited.

Table 2-3 shows the possible meta data.

Name in CloudConn DB	Required?	Juconn name	Contents
`meta_type`	Yes	type	Sensor type
`meta_unit`	No	unit	For the values, see Table 2-4, "Base Unit" column
`meta_setpoint`	No	setpoint	Refers to meta_type
`meta_virtual`	No	virtual	`true` or `false`
`meta_receive_signal`	No	receive_signal	`REGULAR` or `IRREGULAR` Defines whether sensor data is sent regularly
`meta_signal_frequency`	No	signal_frequency	in seconds This value is expected if the sensor data is to be sent regularly
`meta_custom_fields`	No	custom_fields	
`meta_virtual_device`	No	virtual_device	Groups data points to virtual devices

Table 2-3: Overview of meta data

Table 2-4 shows the sensor types supported by Juconn and their units:

Sensor type	Base unit	Alternative units
temperature	°C	°F, °K
humidity	%	-
air pressure	Pa	bar,psi,atm
water pressure	Pa	bar,psi,atm
light	lm	lx
switch	true / false	-
gas content	%	
volume	l	cm^3
speed	km/h	m/sec, miles/h
gps	long /lat	
power	W	mW
current	A	mA
voltage	V	mV
custom *	any	

Table 2-4: Overview of sensor types

* “custom” sensor type: any type; not a base unit. Can be used for CELLID etc. for example.
The value for the “custom” sensor type can be any, e.g. a string or a number in hexadecimal format. Therefore, the value must be transmitted as a string without interpretation.

Please note: Special characters must be displayed in the TiXML display using entities (e.g. the degrees character° must be displayed as `&#xb0;`).

Body example for gatewayInit (JSON):

```
{
     "request_id": "generated string",
     //--------- Start of optional area --------
     "virtual_devices": "1",
     "gateway_details" : "TBD any details",
     //---------- End of optional area ---------
     "controllers" : [
          {
               "name": " Controller 1 name",
               "external_controller_id" : "$controller_name_1",
     //--------- Start of optional area --------
               "custom_fields" : null,
               "timeout" : 360,
        //---------- End of optional area ---------
        "sensors" : [
             {
                  "external_sensor_id": "$sensor_id_1",
                  "name" : $Varname_1,
                       // e.g. $Varname_1 e.g. “Temp Sensor”
                  for controller 1
                  "type" : "temperature",
                  //--------- Start of optional area --------
                  "unit" : "°C",
                  "set_point" : "",
                  "virtual" : false,
                  "virtual_device" : "Heating_1",
                  "receive_signal" : "enum [REGULAR, IRREGULAR]",
                  "signal_frequency" : 300,
                  "custom_fields" : {
                       "plant": "Plant 1",
                       "loop" : "loop 1"
                  }
             //---------- End of optional area ---------
             },
             {
                  "external_sensor_id": “$sensor_id_2”,
                  "name" : $Varname_2,
                       // e.g. $Varname_2 e.g. “Humidity
                  Sensor” for controller 1

                  "type" : "humidity",
                  //--------- Start of optional area --------
                  "unit" : "%",
                  "set_point" : "",
                  "virtual" : false,
                  "virtual_device" : "Heating_2",
                  "receive_signal" : "enum [REGULAR, IRREGULAR]",
                  "signal_frequency" : 300,
                  "custom_fields" : null
                  //---------- End of optional area ---------
             }
        ]
   },
   {
        "name": " Controller 2 name",
        "external_controller_id" : "$controller_name_2",
        "custom_fields" : null,
        "timeout" : 360,
        "sensors" : [
             {
                  "external_sensor_id": "$sensor_id_1",
                  "name" : $Varname_1,
                       // e.g. $Varname_1 e.g. “Temp Sensor”
                  for controller 2

                  "type" : "temperature",
                  //--------- Start of optional area --------
                  "unit" : "°C",
                  "set_point" : "",
                  "virtual" : false,
                  "virtual_device" : "Heating_1",
                  "receive_signal" : "enum [REGULAR, IRREGULAR]",
                  "signal_frequency" : 300,
                  "custom_fields" : {
                       "plant": "Plant 1",
                              "loop" : "loop 1"
                         }
                         //---------- End of optional area ---------
                    },
                    {
                         "external_sensor_id": "$sensor_id_2",
                         "name" : $Varname_2,
                              // e.g. $Varname_2 e.g. “Humidity
                         Sensor” for controller 2

                         "type" : "humidity",
                         //--------- Start of optional area --------
                         "unit" : "%",
                         "set_point" : "",
                         "virtual" : false,
                         "virtual_device" : "Heating_2",
                         "receive_signal" : "enum [REGULAR, IRREGULAR]",
                         "signal_frequency" : 300,
                         "custom_fields" : null
                         //---------- End of optional area ---------
                    }
               ]
          }
     ]
}
```

... or in TiXML:

```
{
   <?xml version="1.0" encoding="UTF-8" ?>
   <request_id _="generated string"/>
   <virtual_devices _="1" />
   <!-- ---------- Start of optional area --------- -->
   <gateway_details>TBD any details</gateway_details>
   <!-- ---------- End of optional area --------- -->
   <controllers>
     <$controller_name_1>
       <custom_fields/>
       <timeout _="360"/>
       <sensors>
         <$sensor_id_1>
           <name _=$Varname_1/>
           <!-- $Varname_1 e.g. “Temp Sensor”for controller_id1 -->
           <type _="temperature"/>
           <!--------- Start of optional area -------->
           <unit _="°C"/>
           <set_point />
           <virtual _="false"/>
           <virtual_device _="Heating_1"/>
           <receive_signal _="enum [REGULAR, IRREGULAR] "/>
           <signal_frequency _="300"/>
           <custom_fields>
              <plant _="Plant 1"/>
              <loop _="loop 1"/>
           </custom_fields>
           <!---------- End of optional area ---------->
         </$sensor_id_1>
         <$sensor_id_2>
           <name _=$Varname_2/>
           <!-- $Varname_2 e.g. “Humidity Sensor”for controller_id1 -->
```

```
          <type _="humidity"/>
          <!--------- Start of optional area -------->
          <unit _="%"/>
          <set_point />
          <virtual _="false"/>
          <virtual_device _="Heating_2"/>
          <receive_signal _="enum [REGULAR, IRREGULAR] "/>
          <signal_frequency _="300"/>
          <custom_fields/>
          <!---------- End of optional area ---------->
        </$sensor_id_2>
      </sensors>
    </$controller_name_1>
    <$controller_name_2>
     <custom_fields />
     <timeout _="360"/>
     <sensors>
        <$sensor_id_1>
          <name _=$Varname_1/>
          <!-- $Varname_1 e.g. “Temp Sensor” for controller_id2 -->

          <type _="temperature"/>
          <!--------- Start of optional area -------->
          <unit _="°C"/>
          <set_point />
          <virtual _="false"/>
          <virtual_device _="Heating_1"/>
          <receive_signal _="enum [REGULAR, IRREGULAR] "/>
          <signal_frequency _="300"/>
          <custom_fields>
             <meta_plant _="Plant 1"/>
             <loop _="loop 1"/>
          </custom_fields>
          <!---------- End of optional area ---------->
       </$sensor_id_1>
       <$sensor_id_2>
         <name _=$Varname_2/>
          <!-- $Varname_2 e.g. “Humidity Sensor” for controller_id2 -->

         <type _="humidity"/>
         <!--------- Start of optional area -------->
         <unit _="%"/>
         <set_point />
         <virtual _="false"/>
         <virtual_device _="Heating_2"/>
         <receive_signal _="enum [REGULAR, IRREGULAR] "/>
         <signal_frequency _="300"/>
         <custom_fields/>
         <!---------- End of optional area ---------->
       </$sensor_id_2>
     </sensors>
    </$controller_name_2>
  </controllers>
}
```

“controllers” and “sensors” must be adapted according to the real layout. The broker therefore takes over the transmitted controllers and sensors, and it ready to receive data.
Controller_id1, controller_id2, sensor_id1 and sensor_id2 are each unique identifiers within the current branch.
In the event of an error, the FP gateway must repeat the rollout process.

2.2.2 Wait for init result

Initialisation is completed by the broker in which the result is sent back to the gateway via the following topic:

Topic

```
$customer_id/fp/$gateway_id/gatewayInitResult
```

Body

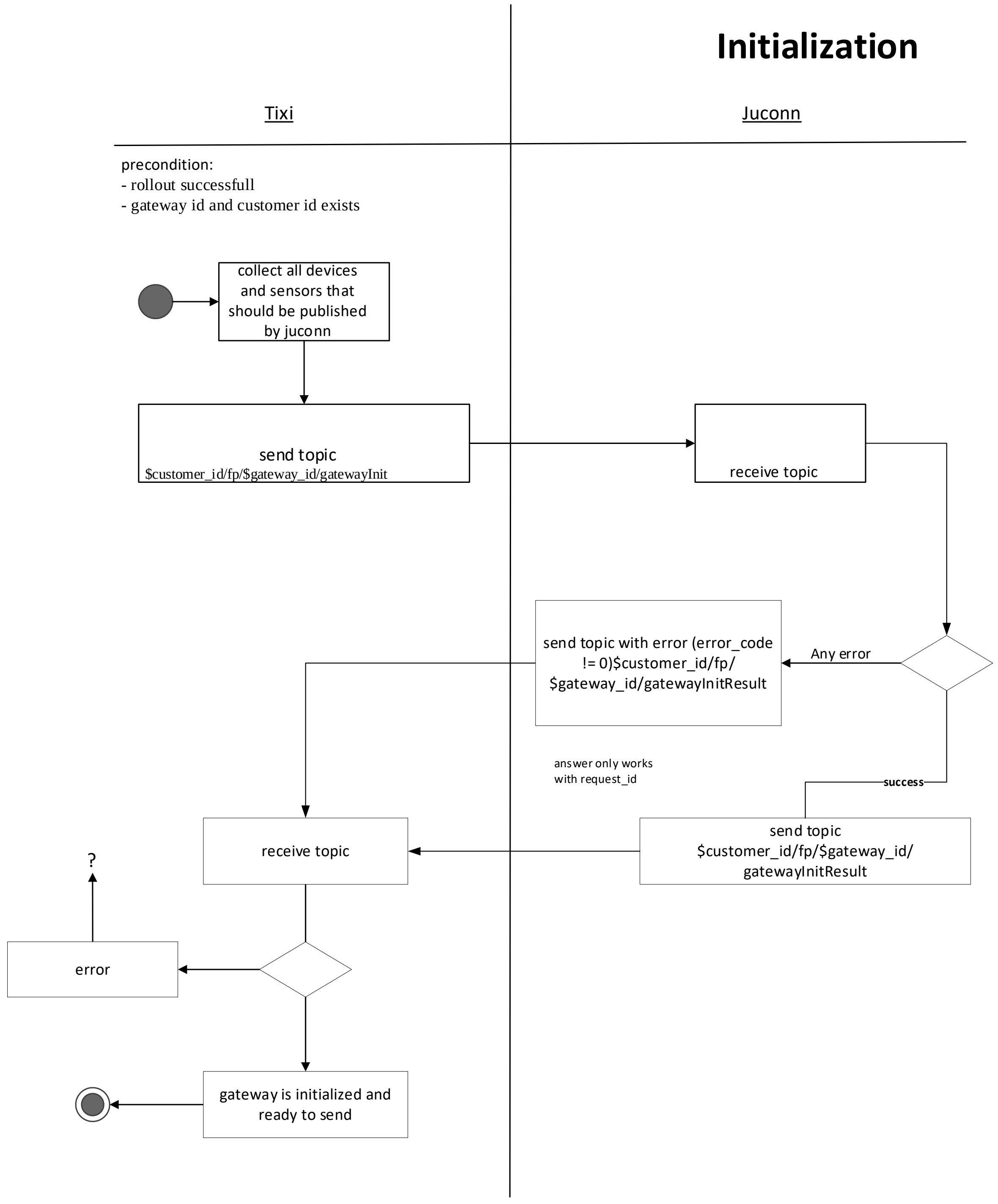
```
{
    "error_code": 0,
    "response_id": "the received request_id"
    "portal_version": "Portal_Version",
}
```

... or in TiXML:

```
{
    <?xml version="1.0" encoding="UTF-8" ?>
    <error_code _="0"/>
    <response_id _="the received request_id"/>
    <portal_version _="Portal_Version" />
}
```

The Portal_Version variable is used to transmit the broker's software version to the gateway. This value is intended to help Support in particular. For information regarding displaying the Portal_Version, see section 7.5.

The gateway is now ready to send sensor data to the broker.


Initialization
Tixi
Juconn
precondition:
- rollout successfull
- gateway id and customer id exists
collect all devices and sensors that should be published by juconn
send topic $customer_id/fp/$gateway_id/gatewayInit
receive topic
send topic with error (error_code != 0)$customer_id/fp/$gateway_id/gatewayInitResult
Any error
answer only works with request_id
success
send topic $customer_id/fp/$gateway_id/gatewayInitResult
receive topic
?
error
gateway is initialized and ready to send

2.3 Running

When sending the data, either the data for an individual sensor, that for an individual controller or data for several / all controllers can be sent.

The message_id must represent an increasing decimal and is used by the broker to ignore data that has been sent twice.
When `$customer_id/fp/$gateway_id/gatewayInitResult` is received, the message_id can
be set to zero.

2.3.1 Sending data for one sensor

This topic transmits the value for one sensor.
The option to send individual data points is used if

a) The realtime section within the CloudConn database only contains one data point
b) Only one data point is to be sent currently when using the hysteresis function

Topic

$customer_id/fp/$gateway_id/data

Body

```
{
    "message_id": 1.12,
    "timestamp": 1498724247,
    "controllers": [
        {
            "external_controller_id": "$controller_name_1",

            "sensors": [
                {
                    "external_sensor_id": "$sensor_id_1",
                    "value": "any value object",
                    "sensor_state": 0
                }
            ]
        }
    ]

}
```

… or in TiXML:

```
{
    <?xml version="1.0" encoding="UTF-8" ?>
    <message_id _="1.12"/>
    <timestamp _="1498724247"/>
    <controllers>
       <$controller_name_1>
         <sensors>
          <$sensor_id_1 _="1" state="0"/>
         </sensors>
       </$controller_name_1>
    </controllers>
}
```

The time stamp `timestamp` is the Unix time (epoch). The time zone is UTC unless otherwise configured in the CloudConn database.

The TiXML definition and payload for "gps" sensors is described in section 7.1. The "`sensor_state`" provides information regarding the variable type and partly regarding its status.
See Table 2-5.

Values for sensor_state	Status description
0 or 1	indicates the DeviceState for a bus variable 0 = data invalid; 1 = data valid
2	indicates that this is a process variable, a variable from an expansion module or a variable with position data (GNSS / GPS -> sensor type=gps), for which there is no DeviceState

Table 2-5: Encoding the sensor status

2.3.2 Sending data for one controller

This topic transmits all sensor values for one controller.
The option to send individual data points is used if

a) The realtime section within the CloudConn database only contains data points from one controller. At least 2 data points have to be sent.
b) Only data points from one controller are to be sent currently and at least 2 data points are to be sent when using the hysteresis function.

Topic

$customer_id/fp/$gateway_id/data

Body

```
{
    "message_id": 1.12,
    "timestamp": 1498724247,
    "controllers": [
        {
            "external_controller_id": "$controller_name_1",

            "sensors": [
                {
                    "external_sensor_id": "$sensor_id_1",
                    "value": "any value object",
                    "sensor_state": 0
                },
                {
                    "external_sensor_id": "$sensor_id_2",
                    "value": "any value object",
                    "sensor_state": 1
                },
                {
                     "external_sensor_id": "$sensor_id_3",
                     "value": "any value object",
                     "sensor_state": 2
                }
           ]
        }
    ]
}
```

... or in TiXML:

```
{
    <?xml version="1.0" encoding="UTF-8" ?>
    <message_id _="1.12"/>
    <timestamp _="1498724247"/>
    <controllers>
       <$controller_name_1>
         <sensors>
          <$sensor_id_1 _="1" state="0"/>
          <$sensor_id_2 _="2" state="1"/>
          <$sensor_id_3 _="5201.4613 N" lng="14208.4613 E" state="2"/>
         </sensors>
       </$controller_name_1>
    </controllers>
}
```

In the example shown above, id_3 is a type "gps" sensor (special display).
The TiXML definition and payload for "gps" sensors is described in section 7.1.

The time stamp `timestamp` is the Unix time (epoch). The time zone is UTC unless otherwise configured in the CloudConn database.

2.3.3 Sending data for several controllers

This topic transmits the sensor values for all controllers to the broker.
Sending “several controllers” is the default sending method and suitable for all purposes

Topic

$customer_id/fp/$gateway_id/data

Body

```
{
    "message_id": 1.12,
    "timestamp": 1498724247,
    "controllers": [
        {
            "external_controller_id": "$controller_name_1",
            "sensors": [
                {
                    "external_sensor_id": "$sensor_id_1",
                    "value": "any value object",
                    "sensor_state": 0
                },
                {
                    "external_sensor_id": "$sensor_id_2",
                    "value": "any value object",
                    "sensor_state": 1
                }
            ]
        },
        {
            "external_controller_id": "$controller_name_2",
            "sensors": [
                {
                    "external_sensor_id": "$sensor_id_1",
                    "value": "any value object",
                    "sensor_state": 0
                },
                {
                    "external_sensor_id": "$sensor_id_2",
                    "value": 23.9,
                    "sensor_state": 0
                }
            ]
        }
    ]
}
```

Or in TiXML

```
{
   <?xml version="1.0" encoding="UTF-8" ?>
   <message_id _="1.12"/>
   <timestamp _="1498724247"/>
   <controllers>
      <$controller_name_1>
        <sensors>
          <$sensor_id_1 _="1" state="0"/>
          <$sensor_id_2 _="2" state="1"/>
          <$sensor_id_3 _="5201.4613 N" lng="14208.4613 E" state="2"/>
        </sensors>
      </$controller_name_1>

      <$controller_name_2>
        <sensors>
          <$sensor_id_1 _="3" state="0" />
          <$sensor_id_2 _="4" state="0" />
          <$sensor_id_3 _="5" state="1" />
          <$sensor_id_4 _="6" state="2" />
        </sensors>
      </$controller_name_2>
   </controllers>
}
```

The time stamp `timestamp` is the Unix time (epoch). The time zone is UTC unless otherwise configured in the CloudConn database.

In the example shown above, id_3 is a type “gps” sensor (special display).
The TiXML definition and payload for “gps” sensors is described in section 7.1.

3 Sending config data

Configuration parameters can be sent from the broker to the FP gateway using the following topic. It should be possible to send both SetConfig commands and set commands.

Topic

$customer_id/fp/$gateway_id/config

Body

```
{
    "request_id": "generated string",
    "serial_number": "$external_gateway_serial_number",
    "config_xml": "could only contain statements beginning with SetConfig"
}
```

Or in TiXML

```
{
    <?xml version="1.0" encoding="UTF-8" ?>
    <request_id _="generated string"/>
    <serial_number _="$external_gateway_serial_number" />
    <config>
       could only contain statements beginning with SetConfig
    </config>
}
```

Example 1 for a config_xml `SetConfig` statement:

```
    <SetConfig _="USER" ver="y">
     <AccRights>
               <Groups>
               <Admin>
                    <LocalLogin AccLevel="1"/>
                   <CardLogin AccLevel="1"/>
                   <RemoteLogin AccLevel="1"/>
                   <EthernetLogin AccLevel="1"/>
               </Admin>

               <Webserver>
                    <WebServer AccLevel="20"/>
               </Webserver>
          </Groups>

            <User _="Plain">
                    <Def_LocalLogin Plain="" Group="Admin"/>
                    <Def_CardLogin Plain="" Group="Admin"/>
                    <Def_RemoteLogin Plain="" Group="Admin"/>
                    <Def_EthernetLogin Plain="" Group="Admin"/>
                    <ADMIN Plain="" Group="Admin"/>
            </User>
          </AccRights>
    </SetConfig>
```

Example 2 for a config_xml `Set` statement:

```
<Set _="/Process/PV/Test" value="10" ver="y">
```

If the serial number does not match that saved in the device or if the SetConfig command contains errors, there is a corresponding error code in the subsequent response. The SetConfig command is only executed if the serial_number matches that saved in the device.

The gateway completes configuration using:

Topic

$customer_id/fp/$gateway_id/configResult

Body

```
{
   "error_code": 0,
   "response_id": "the request id for the config",
   "config_result_xml": "string with the result",
}
```

… or in TiXML:

```
{
   <?xml version="1.0" encoding="UTF-8" ?>
   <error_code _="0"/>
   <response_id _="the request id for the config"/>
   <config_result_xml _="string with the result"/>
}

Or

{
   <?xml version="1.0" encoding="UTF-8" ?>
   <error_code _="0"/>
   <response_id _="c30507d0-0618-11ea-8009-efa48641463e"/>
   <config_result_xml>
     <"string with the result">
   </config_result_xml>
}
```

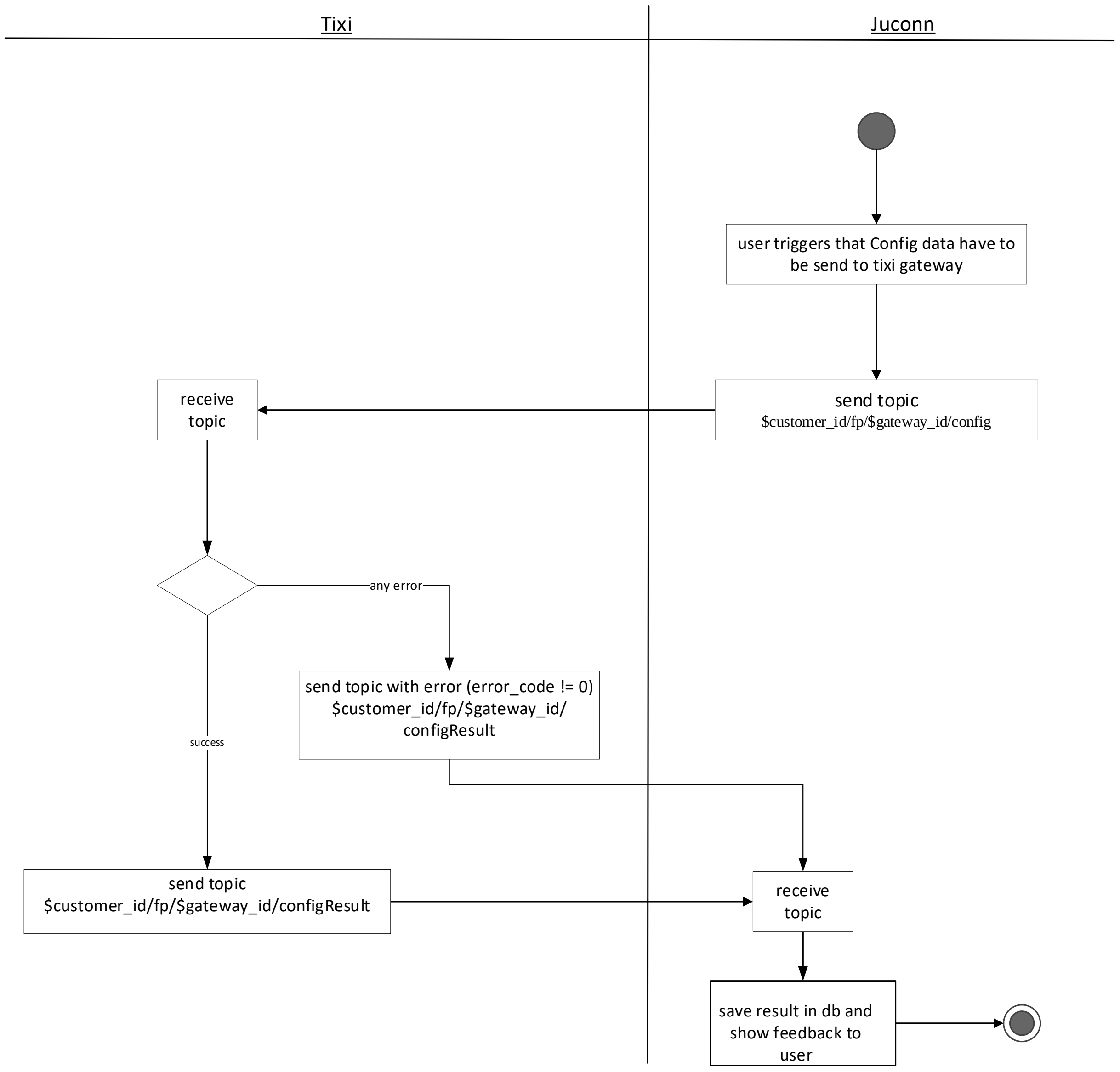
An incorrectly executed <SetConfig> … </SetConfig> job is responded to with <SetConfig/>.

Important:
Before each SetConfig configuration command (i.e. all `SetConfig` commands; not valid for `Set` commands !), internal processing should always be stopped using the following command:

```
<Set _="/Process/Program/Mode" value="Stop" ver="v"/>
```

Wait for the result of the aforementioned command. A `SetConfig` command should only be executed once the aforementioned stop command has been executed successfully.

Sending Config Data from Juconn to Tixi Gateway


Tixi
Juconn
user triggers that Config data have to be send to tixi gateway
send topic $customer_id/fp/$gateway_id/config
receive topic
any error
send topic with error (error_code != 0) $customer_id/fp/$gateway_id/ configResult
success
send topic $customer_id/fp/$gateway_id/configResult
receive topic
save result in db and show feedback to user

Depending on the SetConfig command, it can take between a few seconds (e.g. USER database) and several hours (M-bus scan command) until the gateway responds to the command.
The broker can decide how it wishes to continue.

4 Sending command to FP gateway

The MQTT broker can use a feedback channel to send any TiXML commands to the FP gateway in order to save a new configuration to the device, to restart the device or to switch an output for example. The MQTT broker can send all TiXML commands that are also possible using the TICO configuration software to the device.
The client subscribes to the following topic from the broker:

Topic

```
$customer_id/fp/$juconn_gateway_id/cmd
```

Body

```
{
    "request_id": "generated string",
    "serial_number": "$external_gateway_serial_number",
    "cmd": {
             any command
            }
}
```

... or in TiXML:

```
{
    <?xml version="1.0" encoding="UTF-8" ?>
    <request_id _=”generated string”/>
    <serial_number _=$external_gateway_serial_number/>

    <cmd>
             any command
    </cmd>
}
```

Example:

```
{
    "request_id": "generated string",
    "serial_number": "$external_gateway_serial_number",
    "cmd": {
     "SetConfig": [
        {
           "-_": "USER",
           "-ver": "y",
           "AccRights": {
              "Groups": {
                 "Admin": {
                    "CardLogin": {"-AccLevel": 1},
                    "EthernetLogin": {"-AccLevel": 1},
                    "LocalLogin": {"-AccLevel": 1},
                    "RemoteLogin": {"-AccLevel": 1}
                 },
                 "Webserver": {"WebServer": {"-AccLevel": 20}
```

```
                    }
                 },
                 "User": {"-_": "Plain","ADMIN":{"-Plain":"","-Group":"Admin"},
                    "Def_CardLogin": {"-Plain": "", "-Group": "Admin"},
                    "Def_EthernetLogin": {"-Plain": "", "-Group": "Admin"},
                    "Def_LocalLogin": {"-Plain": "", "-Group": "Admin"},
                    "Def_RemoteLogin": {"-Plain": "", "-Group": "Admin" }
                 }
              }
           }
       ]
     }
}
```

Or in TiXML

```
{
    <?xml version="1.0" encoding="UTF-8" ?>
    <request_id _="generated string"/>
    <serial_number _=$external_gateway_serial_number/>

    <cmd>

         <!-- Here comes the TiXML command...
              Example: SetConfig of AccRights Database
         -->

         <SetConfig _="USER" ver="y">
               <AccRights>
                  <Groups>
                     <Admin>
                       <LocalLogin AccLevel="1"/>
                       <CardLogin AccLevel="1"/>
                       <RemoteLogin AccLevel="1"/>
                       <EthernetLogin AccLevel="1"/>
                     </Admin>

                     <Webserver>
                       <WebServer AccLevel="20"/>
                     </Webserver>
                  </Groups>

                  <User _="Plain">
                       <Def_LocalLogin Plain="" Group="Admin"/>
                       <Def_CardLogin Plain="" Group="Admin"/>
                       <Def_RemoteLogin Plain="" Group="Admin"/>
                       <Def_EthernetLogin Plain="" Group="Admin"/>
                       <ADMIN Plain="" Group="Admin"/>
                  </User>
               </AccRights>
            </SetConfig>

         </cmd>
    }
```

The command is only executed if the gateway's serial_number matches.

Possible errors

code 500:
invalid command

Important:
Before each SetConfig configuration command (i.e. all `SetConfig` commands; not valid for any other commands such as `Set` commands, `DoOn,` etc. !), internal processing should always be stopped using the following command:

```
<Set _="/Process/Program/Mode" value="Stop" ver="v"/>
```

Wait for the result of the aforementioned command. A `SetConfig` command should only be executed once the aforementioned stop command has been executed successfully.

The client sends the command processing result back to the broker using a publish topic:

Topic

```
$customer_id/fp/$juconn_gateway_id/cmdResult
```

Body

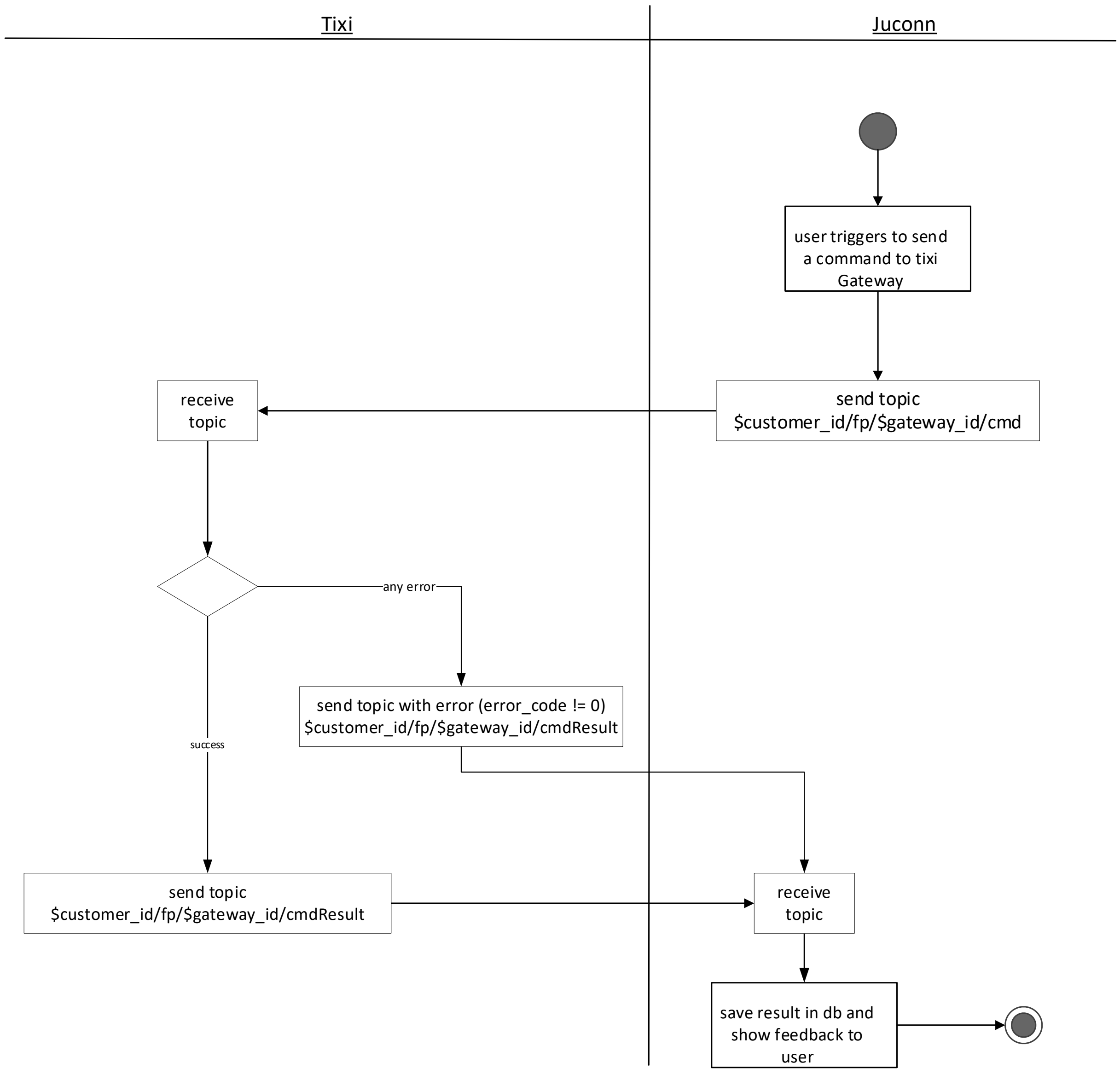
```
{
    "error_code": 0,
    "response_id": "the request id for the config",
    "cmd_result":
    {
     "SetConfig": null
    }
}
```

... or in TiXML:

```
{
    <?xml version="1.0" encoding="UTF-8" ?>
    <error_code _="0"/>
    <response_id _="the request id for the config"/>
    <cmd_result>
     <!--     Here comes the command response from FP device.
              The response is exactly what you see in TICO, except the
              command frame ([ ] will be omitted).
     -->

     <!--     Here comes the TiXML command response ...
              Example: SetConfig of AccRights Database was sent
              Response is an empty SetConfig command or an error
              description, if the command execution failed.
        -->
     <SetConfig/>
    </cmd_result>
}
```

Sending Cmd from Juconn to Tixi Gateway


Tixi
Juconn
user triggers to send a command to tixi Gateway
send topic $customer_id/fp/$gateway_id/cmd
receive topic
any error
send topic with error (error_code != 0) $customer_id/fp/$gateway_id/cmdResult
success
send topic $customer_id/fp/$gateway_id/cmdResult
receive topic
save result in db and show feedback to user

5 Send status request to FP gateway

The broker can use a status request to request gateway-specific information from the gateway, e.g.

- Mobile communications data (IMEI, IMSI, provider, reception strength, service, etc.)
  This data is only available if the gateway has a mobile communications module.
- Hardware information (flash memory size, RAM, HW revision, MAC address)
- Available I/Os (onboard interfaces, expansion modules, etc.)
- System information (CPU load, board temperature, backup battery charge level, WiFi status)
- Time information (time, last shutdown, last startup, cycle times)
- Software information (version of Uboot/Linux/Linux-App/configuration)

The following topic is used to collect all of the aforementioned status information and to send it back to the broker.

Topic

$customer_id/fp/$juconn_gateway_id/state

Body

```
{
    "request_id": "generated string",
    "serial_number": "$external_gateway_serial_number",
}
```

… or in TiXML:

```
{
    <?xml version="1.0" encoding="UTF-8" ?>
    <request_id _=”generated string“/>
    <serial_number _=$external_gateway_serial_number />
    <!-- optional path. Only the "Ethernet" path should be sent -->
    <path _="Ethernet" />
}
```

The FP gateway responds to this request with:

Topic

$customer_id/fp/$juconn_gateway_id/stateResult

Body

```
{
    "error_code": 0
    "response_id": " the request id of the status request",
    "gateway_states": {
        "gateway_state"
    }
}
```

Or in TiXML

```
{
    <?xml version="1.0" encoding="UTF-8" ?>
    <response_id _="generated string"/>
    <serial_number _="$external_gateway_serial_number" />
    <error_code> _="0"/>
    <gateway_states>
        "gateway_state"
    </gateway_states>
}
```

The `gateway_state` supplies the response to the TiXML command `<Get ver="y" />`.
If the path parameter is also sent, the gateway should consider this path in the get command: `<Get _="path" ver="y" />`
Example: `path _="Ethernet": <Get _="Ethernet/" ver="y" />`

Example:

```
{
"error_code": 0
"response_id": " the request id of the status request",
"gateway_states": {
{
  "Get": {
    "SYSTEM": {
      "Hardware": {
        "Modules": {
          "RTC": { "-_": "RTC8564" },
          "Modem0": { "-_": "WT640" },
          "GsmModule": { "-_": "Huawei ME909s-120" },
          "ModuleVer": { "-_": "11.617.09.00.00" },
          "FlashOnboard": { "-_": "128MB" },
          "COM1": { "-_": "RS232" },
          "COM2": { "-_": "RS485" },
          "COM3": { "-_": "MBus" },
          "ETH1": { "-_": "ETH1" },
          "MainBoardDigital": { "-_": "GP22D-I/O" },
          "C218": { "-_": "S1-AA2" },
          "C318": { "-_": "S1-AA2" },
          "C21a": { "-_": "S1-AA2" },
          "C31a": { "-_": "S1-AA2" },
          "C440": { "-_": "S1-D30G" },
          "C540": { "-_": "S1-WL2" },
          "C640": { "-_": "S1-D30G" }
        },
        "RAM": {
          "Size": { "-_": "128294912" },
          "Attributes": { "-_": "0" }
        },
        "ROM": {
          "Size": { "-_": "134217728" },
          "Attributes": { "-_": "0" }
        },
        "FileSystem": {
          "Size": { "-_": "100663296" },
          "Type": { "-_": "3" },
          "Attributes": { "-_": "0" }
        }
      },
      "Firmware": {
        "Version": { "-_": "5.2.6.38" },
        "Date": { "-_": "2019-04-17 09:17:34" }
      },
      "Linux": {
        "SystemInfo": {
          "-_": "Linux AT91SAM9 2.6.39 #24072547 Rev. 3915 PREEMPT Thu Oct
25 2018 11:40 armv5tejl GNU/Linux
"
        },
        "Uboot": { "-_": "2010.06-svn801 (Dec 07 2013 - 11:56:40) " }
      },
      "LicenseRef": {
        "HW_Rev": { "-_": "BEG653-SC-V11_D6-D40-50-V31" },
        "Oem": { "-_": "FP InovoLabs GmbH" },
        "OName": { "-_": "WT640" },
        "PClass": { "-_": "Wand.Box WT640" },
        "ProdName": { "-_": "WT640" },
        "LicenseID": { "-_": "000100" },
        "ProductID": { "-_": "3030" },
        "UDID": { "-_": "Tixi.com GM20-S1F2K-120 70426-04600912" }
      },
      "EEProm": {
        "LED0": { "-_": "9" }
      },
      "Process": {
        "OneWire": {
          "Mainboard": {
            "DeviceState": { "-_": "1" },
            "ChangeToggle": { "-_": "0" },
            "CPUTemp": { "-_": "33.125" },
            "ExternalPower": { "-_": "1" }
          },
          "Active": { "-_": "1" },
          "FreeMem": { "-_": "4191229" }
        },
        "VPN": {
          "ConnectionState": { "-_": "0" }
        },
        "HTTPConn": "
              ",
        "CloudConn": {
          "ConnectionType": { "-_": "mqtt+ssl:nb-
iot.ram.m2m.telekom.com:8883" },
          "ConnectionState": { "-_": "1" },
          "ConnectionStateMsg": { "-_": "connected" }
        },
        "CloudConn2": "
              ",
        "CloudConn3": "
              ",
        "IBMConn": "
              ",
        "PV": {
          "Alarm_Id": { "-_": "123" },
          "TEMP_OFFICE": { "-_": "23" },
          "TEMP_CPU": { "-_": "49" },
          "HUMIDITY": { "-_": "60" },
          "CURRENT_L1": { "-_": "15" },
          "CURRENT_L2": { "-_": "9" },
          "CURRENT_L3": { "-_": "14" },
          "VOLTAGE_L1": { "-_": "210" },
          "VOLTAGE_L2": { "-_": "223" },
          "VOLTAGE_L3": { "-_": "232" },
          "Alarm_Critical": { "-_": "0" },
          "Alarm_Major": { "-_": "0" },
          "Alarm_Minor": { "-_": "0" },
          "Alarm_Warning": { "-_": "0" },
          "FirstCyclePV": { "-_": "0" },
          "FirstCycleDelayed": { "-_": "0" },
          "Seconds": { "-_": "29" },
          "IsOnline": { "-_": "0" },
          "Quality": { "-_": "-113.00" },
          "QualityOK": { "-_": "0" },
          "CPULoad": { "-_": "91" },
          "GprsConnected": { "-_": "0" }
        },
        "Program": {
          "Mode": { "-_": "Run" }
        },
        "MB": {
          "IO": {
            "I": {
              "P0": { "-_": "1" },
              "P1": { "-_": "1" }
            },
            "IB": {
              "P0": { "-_": "3" }
            },
            "IW": {
              "P0": { "-_": "3" }
            },
            "ID": {
              "P0": { "-_": "3" }
            },
            "Q": {
              "P0": { "-_": "0" },
              "P1": { "-_": "0" }
            },
            "QB": {
              "P0": { "-_": "0" }
            },
            "QW": {
              "P0": { "-_": "0" }
            },
            "QD": {
              "P0": { "-_": "0" }
            }
          },
          "FirstCycle": { "-_": "0" },
          "PollButton": { "-_": "0" },
          "ModemOffHook": { "-_": "0" },
          "MaxCycleTime": { "-_": "142" },
          "CycleTime": { "-_": "1" },
          "SignalLED": { "-_": "9" },
          "Mount": { "-_": "0" },
    "CPULoad": { "-_": "91" },
    "TixmlQueue": { "-_": "1" }
  },
  "SysIO": {
    "P0": { "-_": "1" },
    "P1": { "-_": "0" },
    "P2": { "-_": "0" },
    "P3": { "-_": "1" },
    "P4": { "-_": "4058" }
  },
  "C218": {
    "AO": {
      "P0": { "-_": "0" }
    }
  },
  "C318": {
    "AO": {
      "P0": { "-_": "0" }
    }
  },
  "C21a": {
    "AO": {
      "P0": { "-_": "0" }
    }
  },
  "C31a": {
    "AO": {
      "P0": { "-_": "0" }
    }
  },
  "C440": {
    "I": {
      "P0": { "-_": "0" },
      "P1": { "-_": "0" },
      "P2": { "-_": "0" }
    },
    "IB": {
      "P0": { "-_": "0" }
    },
    "IW": {
      "P0": { "-_": "0" }
    },
    "ID": {
      "P0": { "-_": "0" }
    }
  },
  "C540": {
    "Q": {
      "P0": { "-_": "0" },
      "P1": { "-_": "0" }
    },
    "QB": {
      "P0": { "-_": "0" }
    },
    "QW": {
      "P0": { "-_": "0" }
    },
    "QD": {
      "P0": { "-_": "0" }
    }
  },
  "C640": {
    "I": {
          "P0": { "-_": "0" },
          "P1": { "-_": "0" },
          "P2": { "-_": "0" }
        },
        "IB": {
          "P0": { "-_": "0" }
        },
        "IW": {
          "P0": { "-_": "0" }
        },
        "ID": {
          "P0": { "-_": "0" }
        }
      },
      "COM1PollActive": { "-_": "0" },
      "COM2PollActive": { "-_": "0" },
      "COM3PollActive": { "-_": "0" },
      "COM4PollActive": { "-_": "0" },
      "ETHPollActive": { "-_": "0" }
    },
    "LogCounter": {
      "JobReport": { "-_": "0" },
      "Event": { "-_": "61" },
      "Login": { "-_": "1" },
      "IncomingMessage": { "-_": "0" },
      "FailedIncomingCall": { "-_": "0" },
      "SupportLog": { "-_": "35299" },
      "FatalSystemError": { "-_": "0" }
    },
    "GSM": {
      "SM": "
            ",
      "FD": "
            ",
      "State": { "-_": "missing SIM Card" }
    },
    "TIMES": {
      "TIME": { "-_": "08:10:29" },
      "DATE": { "-_": "2019/05/17" },
      "RFC822DATE": { "-_": "Fri, 17 May 19 08:10:29 +0100" },
      "ISO8601DATE": { "-_": "2019-05-17T07:10:29Z" },
      "PowerOffTime": { "-_": "2019/04/26,11:01:16" },
      "PowerOnTime": { "-_": "2019/04/26,11:09:36" },
      "DAYOFWEEK": { "-_": "Fri" },
      "DAYOFWEEKNO": { "-_": "5" },
      "YYYY_MM_DD": { "-_": "2019_05_17" },
      "HH_MM_SS": { "-_": "08_10_29" },
      "HEXDATE": { "-_": "5CDE6C75" },
      "YYMM": { "-_": "1905" }
    },
    "Ethernet": {
      "Link": { "-_": "100" },
      "LinkState": { "-_": "1" },
      "AssignedIP": { "-_": "193.101.167.163" },
      "SubnetMask": { "-_": "255.255.255.128" },
      "MAC": { "-_": "00:11:E8:5B:B1:20" },
      "Gateway": { "-_": "193.101.167.193" },
      "DNS_1": { "-_": "193.101.167.2" }
    },
    "Ethernet2": "
        ",
    "WLAN": "
        ",
      "GPRSLinkState": { "-_": "0" },
      "FreeFileSize": { "-_": "89587712" },
      "FreeRAMSize": { "-_": "40030208" },
      "PNP_String": { "-_": "Wand.Box WT640" },
      "FeatureList": {
        "-_": "Debug,
Default,
TSAdapter,
POP3 Client,
Time Client,
URL Send,
FTP Send,
Secure FTP Send,
SMTP Client,
CGI DoOn,
HTTP Server In,
HTTP Server Out,
Script Send,
Express E-mail Send,
Express E-mail Recv,
Incoming Call,
Auto Transmode,
SMS Receive,
Job Result Processor,
Remote ModemMode"
      },
      "SerialNo": { "-_": "04600912" },
      "HardwareID": { "-_": "GM20-S1F2K-120" },
      "Components": { "-_": "RTC=RTC8564;Modem0=WT640;GsmModule=Huawei
ME909s-
120;ModuleVer=11.617.09.00.00;FlashOnboard=128MB;COM1=RS232;COM2=RS485;COM3
=MBus;ETH1=ETH1;MainBoardDigital=GP22D-I/O;C218=S1-AA2;C318=S1-AA2;C21a=S1-
AA2;C31a=S1-AA2;C440=S1-D30G;C540=S1-WL2;C640=S1-D30G" }
    }
  }
}
}
}
```

… or in TiXML:

```
{
     <?xml version="1.0" encoding="UTF-8" ?>
     <error_code _="0"/>
     <response_id _="the request id of the status request"/>
     <gateway_states>
<Get>
     <SYSTEM>
          <Hardware>
               <Modules>
                    <RTC _="RTC8564"/>
                    <Modem0 _="WT640"/>
                    <GsmModule _="Huawei ME909s-120"/>
                    <ModuleVer _="11.617.09.00.00"/>
                    <FlashOnboard _="128MB"/>
                    <COM1 _="RS232"/>
                    <COM2 _="RS485"/>
                    <COM3 _="MBus"/>
                    <ETH1 _="ETH1"/>
                    <MainBoardDigital _="GP22D-I/O"/>
                    <C218 _="S1-AA2"/>
                    <C318 _="S1-AA2"/>
                    <C21a _="S1-AA2"/>
                    <C31a _="S1-AA2"/>
                    <C440 _="S1-D30G"/>
                    <C540 _="S1-WL2"/>
                    <C640 _="S1-D30G"/>
               </Modules>
               <RAM>
                    <Size _="128294912"/>
                    <Attributes _="0"/>
               </RAM>
               <ROM>
                    <Size _="134217728"/>
                    <Attributes _="0"/>
               </ROM>
               <FileSystem>
                    <Size _="100663296"/>
                    <Type _="3"/>
                    <Attributes _="0"/>
               </FileSystem>
          </Hardware>
          <Firmware>
               <Version _="5.2.6.38"/>
               <Date _="2019-04-17 09:17:34"/>
          </Firmware>
          <Linux>
               <SystemInfo _="Linux AT91SAM9 2.6.39 #24072547 Rev. 3915
PREEMPT Thu Oct 25 2018 11:40 armv5tejl GNU/Linux&#xa;"/>
               <Uboot _="2010.06-svn801 (Dec 07 2013 - 11:56:40) "/>
          </Linux>
          <LicenseRef>
               <HW_Rev _="BEG653-SC-V11_D6-D40-50-V31"/>
               <Oem _="FP InovoLabs GmbH"/>
               <OName _="WT640"/>
               <PClass _="Wand.Box WT640"/>
               <ProdName _="WT640"/>
               <LicenseID _="000100"/>
               <ProductID _="3030"/>
               <UDID _="Tixi.com GM20-S1F2K-120 70426-04600912"/>
          </LicenseRef>
          <EEProm>
               <LED0 _="9"/>
          </EEProm>
          <Process>
               <OneWire>
                    <Mainboard>
                         <DeviceState _="1"/>
                         <ChangeToggle _="0"/>
                         <CPUTemp _="33.125"/>
                         <ExternalPower _="1"/>
                    </Mainboard>
                    <Active _="1"/>
                    <FreeMem _="4191229"/>
               </OneWire>
               <VPN>
                    <ConnectionState _="0"/>
               </VPN>
               <HTTPConn>
               </HTTPConn>
               <CloudConn>
                    <ConnectionType _="mqtt+ssl:nb-
iot.ram.m2m.telekom.com:8883"/>
                    <ConnectionState _="1"/>
                    <ConnectionStateMsg _="connected"/>
```

```
</CloudConn>
<CloudConn2>
</CloudConn2>
<CloudConn3>
</CloudConn3>
<IBMConn>
</IBMConn>
<PV>
     <Alarm_Id _="123"/>
     <TEMP_OFFICE _="23"/>
     <TEMP_CPU _="49"/>
     <HUMIDITY _="60"/>
     <CURRENT_L1 _="15"/>
     <CURRENT_L2 _="9"/>
     <CURRENT_L3 _="14"/>
     <VOLTAGE_L1 _="210"/>
     <VOLTAGE_L2 _="223"/>
     <VOLTAGE_L3 _="232"/>
     <Alarm_Critical _="0"/>
     <Alarm_Major _="0"/>
     <Alarm_Minor _="0"/>
     <Alarm_Warning _="0"/>
     <FirstCyclePV _="0"/>
     <FirstCycleDelayed _="0"/>
     <Seconds _="29"/>
     <IsOnline _="0"/>
     <Quality _="-113.00"/>
     <QualityOK _="0"/>
     <CPULoad _="91"/>
     <GprsConnected _="0"/>
</PV>
<Program>
     <Mode _="Run"/>
</Program>
<MB>
     <IO>
          <I>
               <P0 _="1"/>
               <P1 _="1"/>
          </I>
          <IB>
               <P0 _="3"/>
          </IB>
          <IW>
               <P0 _="3"/>
          </IW>
          <ID>
               <P0 _="3"/>
          </ID>
          <Q>
               <P0 _="0"/>
               <P1 _="0"/>
          </Q>
          <QB>
               <P0 _="0"/>
          </QB>
          <QW>
               <P0 _="0"/>
          </QW>
          <QD>
               <P0 _="0"/>
          </QD>
     </IO>
```

```
    <FirstCycle _="0"/>
    <PollButton _="0"/>
    <ModemOffHook _="0"/>
    <MaxCycleTime _="142"/>
    <CycleTime _="1"/>
    <SignalLED _="9"/>
    <Mount _="0"/>
    <CPULoad _="91"/>
    <TixmlQueue _="1"/>
</MB>
<SysIO>
    <P0 _="1"/>
    <P1 _="0"/>
    <P2 _="0"/>
    <P3 _="1"/>
    <P4 _="4058"/>
</SysIO>
<C218>
    <AO>
        <P0 _="0"/>
    </AO>
</C218>
<C318>
    <AO>
        <P0 _="0"/>
    </AO>
</C318>
<C21a>
    <AO>
        <P0 _="0"/>
    </AO>
</C21a>
<C31a>
    <AO>
        <P0 _="0"/>
    </AO>
</C31a>
<C440>
    <I>
        <P0 _="0"/>
        <P1 _="0"/>
        <P2 _="0"/>
    </I>
    <IB>
        <P0 _="0"/>
    </IB>
    <IW>
        <P0 _="0"/>
    </IW>
    <ID>
        <P0 _="0"/>
    </ID>
</C440>
<C540>
    <Q>
        <P0 _="0"/>
        <P1 _="0"/>
    </Q>
    <QB>
        <P0 _="0"/>
    </QB>
    <QW>
        <P0 _="0"/>
```

```
          </QW>
          <QD>
               <P0 _="0"/>
          </QD>
     </C540>
     <C640>
          <I>
               <P0 _="0"/>
               <P1 _="0"/>
               <P2 _="0"/>
          </I>
          <IB>
               <P0 _="0"/>
          </IB>
          <IW>
               <P0 _="0"/>
          </IW>
          <ID>
               <P0 _="0"/>
          </ID>
     </C640>
     <COM1PollActive _="0"/>
     <COM2PollActive _="0"/>
     <COM3PollActive _="0"/>
     <COM4PollActive _="0"/>
     <ETHPollActive _="0"/>
</Process>
<LogCounter>
     <JobReport _="0"/>
     <Event _="61"/>
     <Login _="1"/>
     <IncomingMessage _="0"/>
     <FailedIncomingCall _="0"/>
     <SupportLog _="35299"/>
     <FatalSystemError _="0"/>
</LogCounter>
<GSM>
     <SM>
     </SM>
     <FD>
     </FD>
     <State _="missing SIM Card"/>
</GSM>
<TIMES>
     <TIME _="08:10:29"/>
     <DATE _="2019/05/17"/>
     <RFC822DATE _="Fri, 17 May 19 08:10:29 +0100"/>
     <ISO8601DATE _="2019-05-17T07:10:29Z"/>
     <PowerOffTime _="2019/04/26,11:01:16"/>
     <PowerOnTime _="2019/04/26,11:09:36"/>
     <DAYOFWEEK _="Fri"/>
     <DAYOFWEEKNO _="5"/>
     <YYYY_MM_DD _="2019_05_17"/>
     <HH_MM_SS _="08_10_29"/>
     <HEXDATE _="5CDE6C75"/>
     <YYMM _="1905"/>
</TIMES>
<Ethernet>
     <Link _="100"/>
     <LinkState _="1"/>
     <AssignedIP _="193.101.167.163"/>
     <SubnetMask _="255.255.255.128"/>
     <MAC _="00:11:E8:5B:B1:20"/>
```

```
                <Gateway _="193.101.167.193"/>
                <DNS_1 _="193.101.167.2"/>
            </Ethernet>
            <Ethernet2>
            </Ethernet2>
            <WLAN>
            </WLAN>
            <GPRSLinkState _="0"/>
            <FreeFileSize _="89587712"/>
            <FreeRAMSize _="40030208"/>
            <PNP_String _="Wand.Box WT640"/>
            <FeatureList
_="Debug,&#xd;&#xa;Default,&#xd;&#xa;TSAdapter,&#xd;&#xa;POP3
Client,&#xd;&#xa;Time Client,&#xd;&#xa;URL Send,&#xd;&#xa;FTP
Send,&#xd;&#xa;Secure FTP Send,&#xd;&#xa;SMTP Client,&#xd;&#xa;CGI
DoOn,&#xd;&#xa;HTTP Server In,&#xd;&#xa;HTTP Server Out,&#xd;&#xa;Script
Send,&#xd;&#xa;Express E-mail Send,&#xd;&#xa;Express E-mail
Recv,&#xd;&#xa;Incoming Call,&#xd;&#xa;Auto Transmode,&#xd;&#xa;SMS
Receive,&#xd;&#xa;Job Result Processor,&#xd;&#xa;Remote ModemMode"/>
            <SerialNo _="04600912"/>
            <HardwareID _="GM20-S1F2K-120"/>
            <Components _="RTC=RTC8564;Modem0=WT640;GsmModule=Huawei ME909s-
120;ModuleVer=11.617.09.00.00;FlashOnboard=128MB;COM1=RS232;COM2=RS485;COM3
=MBus;ETH1=ETH1;MainBoardDigital=GP22D-I/O;C218=S1-AA2;C318=S1-AA2;C21a=S1-
AA2;C31a=S1-AA2;C440=S1-D30G;C540=S1-WL2;C640=S1-D30G"/>
      </SYSTEM>
</Get>
</gateway_states>
}
```

6 Service routing

The connection to the broker is normally established using the active LAN interface.
The TiXML database `ISP/ISP/OUT` can also be used to establish the MQTT connection via a mobile communications connection or a VPN tunnel:

```
[<SetConfig _="ISP/ISP" ver="y">
    <!-- Define communication interface for services -->
    <OUT>
        <SMTP _="MODEM"/>
        <CBIS _="MODEM"/>
        <POP3 _="MODEM"/>
        <URLSend _="MODEM"/>
        <INetTime _="MODEM"/>
        <HTTPConn _="MODEM"/>
        <CloudConn _="Ethernet"/>
        <IBMConn _="MODEM"/>
        <FTPPut _="MODEM"/>
        <SFTPPut _="MODEM"/>
        <VPN _="MODEM"/>
    </OUT>
</SetConfig>]
```

The following options can be used for the MQTT client:

```
Ethernet
MODEM
VPN
```

7 TiXML configuration (CloudConn)

The MQTT client is configured using a TiXML database.

```
[<SetConfig _="ISP" ver="y">
   <CloudConn>
     <!-- Cloud server URL or IP address -->
     <!-- connection without tls/ssl: tcp://URL:1883 -->
     <!-- connection with tls/ssl: ssl://URL:8883 -->
     <CloudBaseUrl _="tcp://URL_or_IP_of_MQTT_broker:1883"/>

     <!-- optional -->
     <username _="user" />
     <password _="password" />

     <!-- optional values
      The gateway_id and customer_id will be sent to the gateway during
           Rollout process.
      However, it might be possible that these values are pre-configured.
      In this case the Rollout process requires a special 'pre-provisioning'
      on the server side.
      -->
     <gateway_id _="gateway_id" />
     <customer_id _="customer_id" />

     <!-- optional: QoS
            valid values are 0, 1, 2
            Default value: 1
            Please note that not all brokers supports QoS 1 and 2
            (e.g. AWS:  QoS 1 only)
      -->
     <QoS _="1" />

      <!-- optional: client authentication setting. Default = hsm
           file = keys and certificates will be stored in internal flash
           memory. Path: /flash_user/app/VPN
           files to be used are defined with TLS__Client_key and
           TLS_Client_cert (see below)
           hsm  = keys and certificates will be stored
           in hardware security
           module (requires paho library patch)
      -->
      <TLS_Client_authmode _="file" />

     <!-- optional: key and cert files -->
     <!-- The following parameters will only be used if
          TLS_Client_authmode was set to "file" -->
      <TLS_Client_key _="mykey.key" />
      <TLS_Client_cert _="myclientcert.crt" />

      <!-- optional: server certificate. Default = cabundle
           none = no server authentication
           cabundle = internal Mozilla root certs bundle file
           file = server certificate will be stored in internal flash
           memory. Path: /flash_user/app/VPN
           TLS_Server_cert (see below)
           hsm  = server certificate will be stored in hardware
           security module (requires paho library patch)
      -->
      <TLS_Server_authmode _="file" />
     <!-- optional: key and cert files -->
     <!-- The following parameters will only be used if
          TLS_Client_authmode was set to "file" -->
      <TLS_Server_cert _="servercert.crt" />

     <!-- if set to 1 the MQTT connection will be started at device
          startup; otherwise MQTT connection will not be established -->
     <CloudConnStart _="1"/>

     <!-- fixed setting (mandatory) -->
     <CloudConnType _="FPIL_Tixi_MQTT_1"/>

     <!-- data format for payload JSON, TiXML or XML (default: TiXML)
          In Version 1 of the client only TiXML is supported -->
     <CloudConnDataFormat _="TiXML"/>

     <!-- optional: MQTT Keep Alive Time in seconds (default: 300) -->
     <MQTTKeepAlive _="200"/>

     <!-- optional: Flag for Cloud: Create virtual devices (default: 0)
          Set to 1 if only virtual devices should be used in the Cloud,
          If set to 1 the tag meta_virtual_device is required for
          *every* data point within *all* Realtime sections -->
     <virtual_devices _="1"/>

     <!-- realtime send interval. range: 1s .. 2^30 s;
          default is 3 seconds
     -->
     <RTSendTime1 _="2" />

     <!-- List of variables which should be published;
          Will be published according to RTSendTime1 -->
     <Realtime1>
       <Datapoint1 _="/Process/Bus1/Device1/DP1" [meta_...]/>
       <Datapoint2 _="/Process/Bus1/Device1/DP2" [meta_...]/>
       <Datapoint3 _="/Process/Bus1/Device2/DP3" [meta_...]/>
       <Datapoint4 _="/Process/PV/ProcVar1" [meta_...]/>
       <Datapoint5 _="/Process/PV/ProcVar2" [meta_...]/>
       <!-- special data type: GNSS
          The dummy attribute value is Latitude, lng value is Longitude
       -->
       <GNSS _="/GNSS/Latitude" lng="/GNSS/Longitude" meta_type="gps" />
     </Realtime1>

     </CloudConn>

</SetConfig>]
```

7.1 Using the “gps” sensor type

Juconn knows the “gps” sensor type in which lng (Longitude) and lat (Latitude) are transmitted.
As FP gateways do not support data types with more than one value, the “gps” type is encoded especially in the CloudConn database and then composed of 2 values when transmitting the payload.

As soon as meta_type="gps" occurs, a second value (lat="") must be defined. The state is set statically to "2" (As for ProcessVars). This value pair must be transmitted together as a payload.

Error handling:
If the gps values cannot be resolved (as they are not present), empty values should be transmitted.

Payload example 1 (values exist):

```
<GNSS>
  <sensors>
     <Longitude _="5229.4129" state="2"/>
     <Latitude _="01323.6341" state="2"/>
  </sensors>
</GNSS>
```

Payload example 2 (values do NOT exist):

```
<GNSS>
  <sensors>
     <Longitude _="" state="2"/>
     <Latitude  _="" state="2"/>
  </sensors>
</GNSS>
```

7.2 Hysteresis

For Timer 1 (`RTsendTime1`), there is the option to set whether each data point is to be transmitted depending on a hysteresis setting.

Example: Data point “Temperature”
Start value = 20.5K
Absolute hysteresis = 0.5

Result: If the data point’s value changes to 20 or 21, this is transmitted.

Regulations:

- All values are transmitted once after restarting the device.
- If the last value that was transmitted changes by the hysteresis amount, the value is transmitted once again.

The required parameters:

`<RTsendTime1 _="-1" />`

The value `-1` means that data is no longer sent cyclically but using the hysteresis parameters that were set.

`<UpdateRate1 _="Poll time" />`

Poll time specifies the frequency with which the values are requested from the process branch. The poll time is specified in seconds. Value range: 1 .. 3600

`Attention:`

The hysteresis function is currently only available for the first timer (`RTsendTime1 and UpdateRate1`). The hysteresis function can only be used for numeric values.

The hysteresis is defined as a floating comma value in the configuration for each realtime value.
Example:

```
hysteresis="1.5"
```

Definition:

- The comma / decimal point is displayed with "."
- Max. 2 decimal places
- No negative numbers

Example:
```
  <RTsendTime1 _="-1" />
  <UpdateRate1 _="20" />

  <Realtime1>
     <Voltage_L1_N _="/Process/Modbus/D1/Voltage_L1_N" hysteresis="2" />
     <CPULoad _="/Process/MB/CPULoad" hysteresis="8.5" />
     <Supply temp _="/Process/PV/Supply temp" hysteresis="0.5" />
     <Return temp _="/Process/PV/Return temp" hysteresis="0.8" />
  </Realtime1>
```

In the example above, the hysteresis is defined for the individual data points.
If the data point `Voltage_L1_N` changes by +/- 2.0 compared to the last value that was transmitted, this value is transmitted. If the data point `CPULoad` changes by +/- 8.5 compared to the last value that was transmitted, this value is transmitted, etc.

7.3 Information regarding realtime sections

The data points that are to be transmitted can be defined in up to 5 realtime sections.

The cycle for each realtime section must be preset.

e.g.
RTSendTime1, …, RTSendTime5.
The data from the corresponding sections <RealTime1>..</RealTime1>, etc. then belongs to this

Example (also see 7.2):
```
<CloudConn>
<RTSendTime1 _="-1" />
<UpdateRate1 _=20 />
<Realtime1>
   <Voltage_L1_N _="/Process/Modbus/D1/Volt_L1_N" hysteresis="2" />
   .
   .
</Realtime1>
<RTSendTime2 _="10" />
<Realtime2>
   <Obis_180 _="/Process/Meter/MT174/Obis_180"/>
</Realtime2>
.
.
<RTSendTime5 _="100" />
<Realtime5>
   <Temp_Board _="/Process/OneWire/Mainboard/TEMP_Board"/>
</Realtime5>
</CloudConn>
```

Data points can also be sent via EventHandler or by command:

Command (TICO, …):
```
    <CloudSendRealtimeData [_="CloudConn[,Realtime_Specifier"]] />
```

CloudConn (optional) = name of the CloudConnector. Values: CloudConn, CloudConn2, CloudConn3. Default: CloudConn
Realtime_Specifier (optional) = name of the realtime section, e.g. Realtime1. Default=Realtime1
If the optional parameter is omitted, all defined sections are sent.

Example 1: Send Realtime1 sections from CloudConn
```
  <CloudSendRealtimeData />
```

Example 2: Send Realtime section "Realtime2" from CloudConn2
```
  <CloudSendRealtimeData _="CloudConn2,Realtime2" />
```

Example 3: Send Realtime section "Realtime3" from CloudConn
```
  <CloudSendRealtimeData _="CloudConn,Realtime3" />
```

7.4 Important information regarding using the MQTT client

Please note the following information:

- The configuration options shown in bold must be configured (mandatory).
- At least one CloudBaseUrl must be configured.
- The path for unencrypted URLs is: **tcp**`://URL-or_IP_of_MQTT_broker:`**1883**
- The path for encrypted URLs is: **ssl**`://URL-or_IP_of_MQTT_broker:`**8883**
- Exactly 1 CloudBaseUrl can be configured
- CloudConnStart and CloudConnType must be configured as described above.
- The data in the Realtime1 branch can be configured freely by the user.

After implementing the MQTT client configuration, the FP gateway must be restarted.

7.5 MQTT connection status display

Status display in the process branch
You can monitor the MQTT client’s connection status in the process branch:
```
[<Get _="/Process/CloudConn/" ver="y" />]
```

Result (example):
```
<Get>
  <CloudConn>
    <ConnectionType _="mqtt+ssl:172.18.121.247:8883"/>
    <ConnectionState _="1"/>
    <ConnectionStateMsg _="connected"/>
    <LastTimeStamp _="1585563331"/>
    <ChangeToggle _="1"/>
    <BrokerVersion _="1.0.0"/>
  </CloudConn>
</Get>
```

`ConnectionType` indicates the type of connection (unencrypted = mqtt/encrypted = mqtt+ssl), the URL / IP address for the broker that is currently active and the MQTT port (e.g. 1883 or 8883).
`ConnectionState` = 0 means: no connection
`ConnectionState` = 1 connection is established
`ConnectionStateMsg` = (registering, not connected, connecting, connected)

`LastTimeStamp` = (time stamp of the last data transmission in Unix time)
`ChangeToggle` = (switches between 0 and 1 each time when the time stamp has changed)
`BrokerVersion` = (version of the broker, see section 2.1.2)

Visual indications via signal LED
The “Signal” LED indicates the MQTT connection status.
Off = no MQTT connection active
Flashing red = MQTT connection is being established
Illuminated green = MQTT connection is established

8 Portal error codes

The current known portal error messages depend on the gateway's state and are listed in Table 8-1.

In state	Value	Meaning
handleGatewayRollout	404	gateway not found
handleGatewayRollout	409	gateway not in rollout state
handleGatewayInit	404	gateway not found
handleGatewayInit	409	gateway not in valid state
	500	response_id bei gatewayInit leer

Table 8-1: Juconn error codes

9 Glossary

An accompanying glossary is enclosed: [2]

CloudConnector	A protocol to connect FP gateways to cloud services
Gateway	In IT, the word gateway [ˈgeɪtweɪ] (entrance and exit) designates a component (Hardware and/or software) that establishes a connection between two systems
JSON	Java Script Object Notation
MQTT	Message Queuing Telemetry Transport (MQTT) is an open messaging protocol for machine-to-machine communication
TiXML	Simple XML Control Protocol, uses Extensible Mark-up Language (XML) XML-compatible, simplified depiction of XML data (saves characters)
URL	Uniform Resource Locator

10 Bibliography

[1] Francotyp Postalia, Ralf Müller, “GatewayBrokerLifeCycle.pdf”.
[2] Juconn, Julian Dawo, “Tixi_WordingDefinition.pdf”.
[3] Juconn, Julian Dawo, „JUC_TixiIngeration_v4.pdf“.
[4] FP PLC-TiXML Manual: “510058920101_XX_FP-PLC-TiXML-Manual_EN.pdf”.
[5] Juconn, Julian Dawo, „Juconn_MqttMessages171123.pdf“.
[6] FP-TiXML-Reference (EN): “510058920001_XX_FP-TiXML-Reference_EN.pdf”.

xx = Revision of the document, starting with 00
For more information and to download documents, see www.inovolabs.com.

11 History

Version	Change
1.3.1	First English issue
1.3.0	New meta data for data points (virtual_devices), text corrections
1.2.9	Corrections in section 7.2
1.2.8	Additions and corrections to section 7.5
1.2.7	- new document name; new cover sheet - Fixes in section 5:response to status request via response_id, not request_id in another location Section 7.2 added Section 7.3 added
1.2.6	- Fixes in section 5:response to status request via response_id, not request_id
1.2.5	- New “custom” sensor type; see Table 2-4 - Updated description in section 3 - Updated description in section 4
1.2.4	- 2.1.1 Corrections to the rollout process (serial number example corrected)
1.2.3	- 2.1.2 Rollout process revised (security vulnerability closed) - Reformatting
1.2.2	- Table 1-1, Table 1-2 corrected - 2.1.1 Start rollout: Request_id for rollout corrected - 2.2.1 Start gatewayInit corrected - 2.3 Running corrected - 3 Sending config data Run/stop example corrected - 8 Portal error codes added - 11 History added Attention: the JSON examples have not been updated/corrected finally