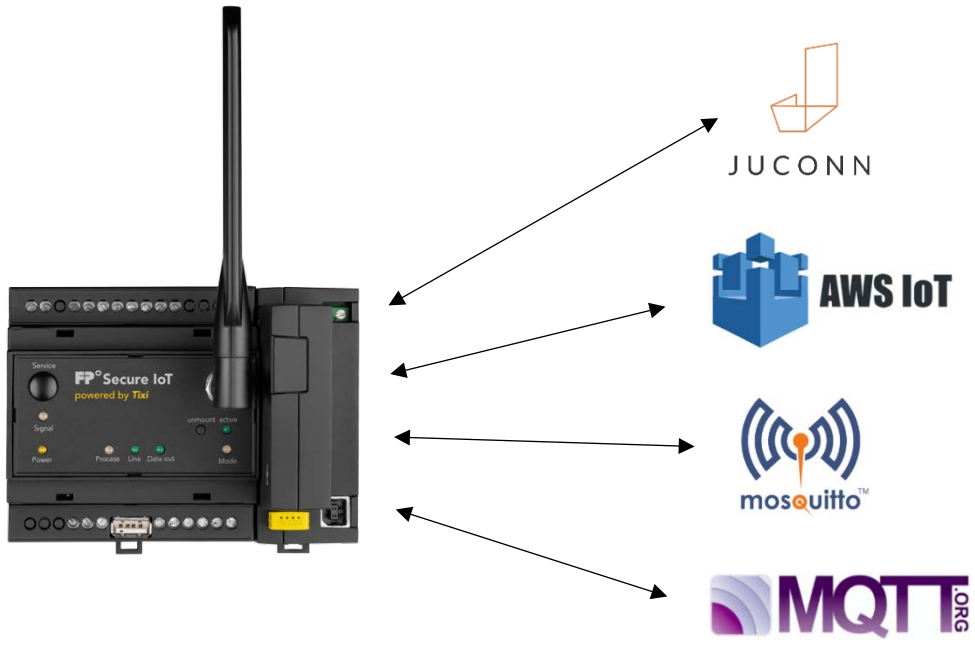
FP Juconn MQTT-Client für FP Gateways Handbuch



Version: 1.3.1

www.inovolabs.com

Redaktionsschluss: 23.02.2021

Inhaltsverzeichnis

1 Einführung

Der FP-Juconn MQTT-Client ist ein universeller MQTT-Client, der für die Juconn-Plattform und AWS gleichermaßen nutzbar ist. Der MQTT Client kann aber prinzipiell auch mit jedem anderen MQTT-Broker verbunden werden, der den aktuellen MQTT Standard „MQTT 3.1.1“ unterstützt.

Rollout, Initialisierung und Normalbetrieb des Gateways

Nach der Ersteinrichtung des Gateways mit einer initialen TiXML-Konfiguration muss das FP Gateway zunächst einmalig einen Rollout-Prozess („gatewayRollout“) mit dem Portal durchlaufen. Beim Rollout-Prozess erhält das Gateway vom Broker einmalig seine eindeutige Gateway ID und Customer ID. Diese beiden IDs werden später als Teil aller Sende-Topics verwendet.

Wenn in der CloudConnector Datenbank die Gateway ID und die Customer ID bereits vordefiniert sind, kann der Rollout auch übersprungen werden.

Nach dem Rollout und bei jedem anschließendem Neustart des Gateways muss eine Initialisierungsroutine („gatewayInit“) durchlaufen werden, in der das Gateway dem Broker sein Datenmodell übermittelt.

Nach erfolgter Initialisierung wechselt das Gateway in den Normalbetrieb (Running-Status) und sendet Daten zum Broker. Weiterhin kann das Gateway Befehle über den Rückkanal empfangen und auswerten.

Sendearten und Sendeintervalle

Sobald eine MQTT-Verbindung aufgebaut wurde (Running-Status), sendet der Client im eingestellten Zyklus Prozessdaten oder Logdaten zum Broker. Optional ist auch die Einrichtung einer Hysterese-Funktion möglich sowie der Ereignis-gesteuerte Versand von Daten. Weiterhin können auch Logdaten versendet werden (z.B. wenn die Kommunikation zum Broker über einen längeren Zeitraum unterbrochen wurde). Gegebenenfalls kann dafür aber auch ein MQTT Caching-Mechanismus verwendet werden.

Die Daten des Gateways können auf folgende Weise zum Portal gesendet werden:

1. Prozessdaten (Process/…)
    a. zyklisch
    b. Ereignis-gesteuert über Sendebefehl „CloudSendRealtimeData“
    c. Hysteresefunktion
2. Logdaten
    a. Zyklisch
    b. Ereignis-gesteuert

Rückkanal zum Steuern des Gateways über den Broker

Es gibt mehrere Rückkanäle, über den der Broker dem Client TiXML-Befehle senden kann.
Die Ergebnisse der Befehle werden dann über einen weiteren Topic an den Broker zurückgesendet (siehe Tabelle 1-1).

Datenformat

Die Daten können jeweils im JSON-Format oder im TiXML-Format verschickt werden. Das Datenformat wird in der CloudConn-Datenbank konfiguriert (**`CloudConnDataFormat`**, siehe Kapitel 7).
In der Version 1 wird nur das TiXML Datenformat unterstützt.

Topics

Die meisten Topics setzen sich aus einem **`festen Teil`** und einem **`variablen Teil`** zusammen. Beispiel: Topic zum Senden aller Daten eines Gateways (publish):
`$customer_id/fp/$gateway_id/data`
Die **`$customer_id`** und die **`$gateway_id`** sind das Ergebnis des Rollout-Prozesses und identifizieren ein Gateway eindeutig innerhalb eines Brokers.

Aus Sicht des Gateways gibt es folgende Topics:

Topic	Richtung	Bedeutung
fp/gatewayRollout	publish	Start Rollout Prozess
fp/gatewayRolloutResult/$serial_number	subscribe	Ergebnis des Rollouts vom Broker empfangen. Es gibt für jeden Gateway einen eigenen Response-Topic (Datensicherheit)
$customer_id/fp/$gateway_id/gatewayInit	Publish	Start Initialization Prozess
$customer_id/fp/$gateway_id/gatewayInitResult	subscribe	Ergebnis des Initialisierungsprozesses vom Broker empfangen
$customer_id/fp/$gateway_id/data	publish	Sende alle Daten Standard-Sende-Topic für Realtime-Daten
$customer_id/fp/$gateway_id/config	subscribe	Gateway empfängt SetConfig-Kommando
$customer_id/fp/$gateway_id/configResult	publish	Gateway sendet Ergebnis des SetConfig-Kommandos
$customer_id/fp/$gateway_id/cmd	subscribe	Gateway empfängt beliebige TiXML-Kommandos
$customer_id/fp/$gateway_id/cmdResult	publish	Gateway sendet Ergebnis des ausgeführten Kommandos
$customer_id/fp/$gateway_id/state	subscribe	Gateway empfängt Kommando zum Senden von allgemeinen Gateway-Informationen
$customer_id/fp/$gateway_id/stateResult	publish	Gateway sendet allgemeine Gateway-Infos an Broker

Tabelle 1-1: Übersicht der verwendeten Topics

Verbindungskontrolle

Im Client kann ein MQTT Broker konfiguriert werden. Der Client versucht dann nach dem Start eine Verbindung zu diesem Broker aufzubauen.

Verschlüsselt vs. unverschlüsselt

Die Verbindung zum Broker kann unverschlüsselt (Port 1883) oder verschlüsselt (Port 8883) aufgebaut werden.

1.1 Inbetriebnahme-Prozess

FP Gateways werden ab Werk mit der passenden Broker-URL voreingestellt.
Werden die Geräte dann zum Kunden geliefert, durchlaufen sie den Inbetriebnahme Prozess mit den Stufen Rollout (optional) und Initialization. Anschließend sind sie in der Lage Daten an den Broker zu verschicken.

Beim Rollout meldet sich das Gateway beim Broker mit seinem Produktnamen und seiner Seriennummer an. In der Antwort des Brokers werden dem Gateway eine $gateway_id, eine $customer_id und ggf. eine Broker-URL für die weitere Kommunikation zugewiesen. Darüber hinaus kann auch eine initiale Konfiguration an das Gateway übermittelt werden.

Damit ist der Rollout-Prozess abgeschlossen. Optional kann der Rollout-Prozess auch übersprungen werden. In diesem Fall werden die $gateway_id und die $customer_id vorkonfiguriert.

Bei der anschließenden Initialisierung teilt das FP Gateway dem Broker mit, welche Sensoren angeschlossen sind (Datenmodell). Dabei werden auch Metadaten wie z.B. Sensortyp, Einheit etc. übermittelt.

Nach Abschluss der Initialisierung sendet das FP Gateway dann die während des Initialisierungsprozesses vereinbarten Daten (z.B. Sensor Daten).

Prozess-Schritte bei der Inbetriebnahme (siehe Tabelle 1-2):
ROLLOUT –> WAIT FOR ROLLOUT RESULT –> INITIALISIERUNG -> WAIT FOR INIT RESULT –> RUNNING

Prozess-Schritt	Zustandsbeschreibung
ROLLOUT (optional)	Anlieferungzustand des Gateways beim Kunden; Juconn Url vordefiniert Gateway meldet sich beim Broker an und sendet seine Seriennummer und den Gerätenamen
WAIT FOR ROLLOUT RESULT (optional)	Das Gateway erwartet erste Informationen vom Broker ($customer_id, $gateway_id; optional Broker-URL und Gateway-Konfiguration)
INITIALISIERUNG	Das Gateway ruft den Topic $customer_id/fp/$gateway_id/gatewayInit auf und sendet seine Konfiguration zum Broker
WAIT FOR INIT RESULT	Gateway wartet auf die Rückmeldung vom Broker
RUNNING	Gateway sendet die Sensordaten; es reagiert auf Broker Kommandos

Tabelle 1-2: Lifecycle Zustandsübersicht

Prozess-Schritt	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	

Tabelle 1-3: Lifecycle Topic Zuordnung

1.2 Datenformat der MQTT-Payload

Das Gateway kann die Payload in den Formaten JSON und TiXML an den Broker senden.
Das Datenformat wird in der CloudConn Datenbank über den Parameter CloudConnDataFormat festgelegt. Standardeinstellung ist TiXML. Das Gateway muss beide Datenformate verarbeiten können.

1.3 Fehlermeldungen des Gateways

Wenn das Gateway Kommandos vom Broker empfängt und diese Kommandos ausführt, werden unter Umständen Fehlermeldungen generiert, wenn das Kommando z.B. falsche Parameter enthält oder aus anderen Gründen nicht ausführbar ist.

Über einen Parameter beim Senden des Kommandos kann dabei spezifiziert werden, wie ausführlich die Fehlermeldungen vom Gateway beantwortet werden.

Es gibt drei mögliche Formate, mit denen diese Fehler mitgeteilt werden. Abhängig von der Vorgabe durch den Parameter „ver“, erfolgt eine kurze oder lange Fehlermeldung. Siehe Tabelle 1-4

Beispielkommando:

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

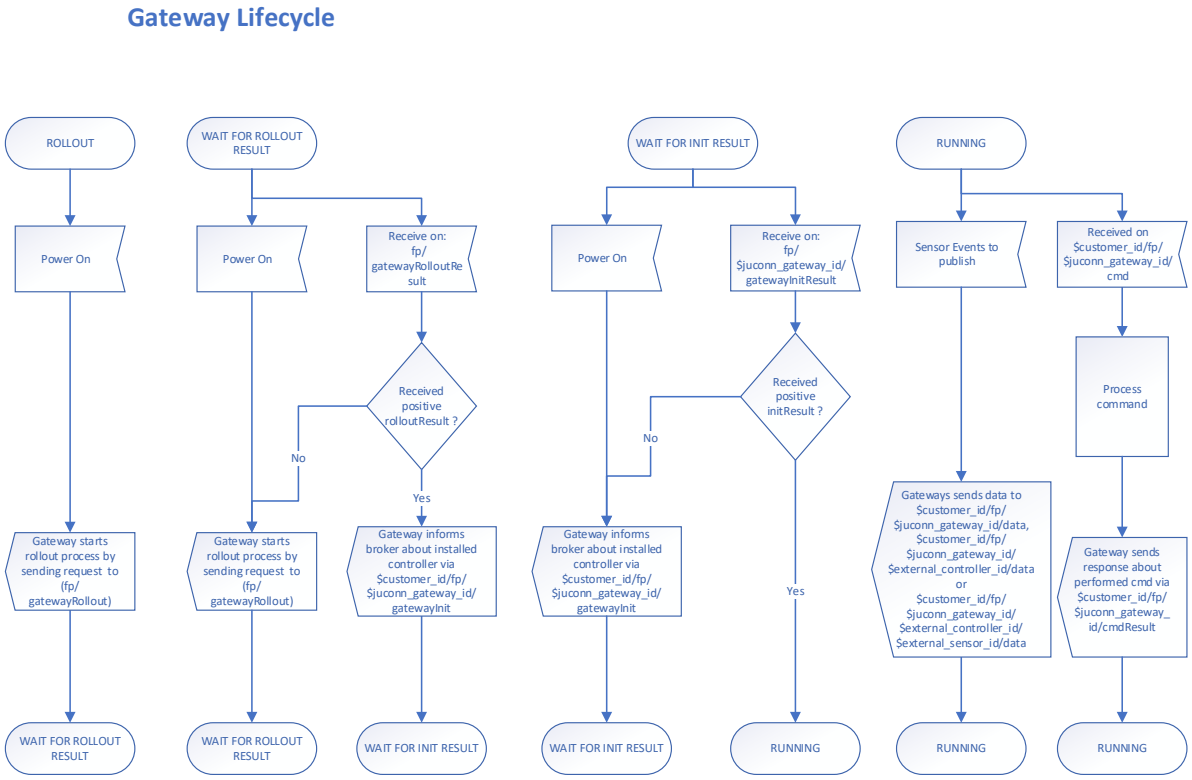
vmode	Beschreibung	Beispiel
n	Fehlermeldung als numerischer Wert	`<Error _="-2196" />`

y	Fehlermeldung, kurze Beschreibung	`<Error>`
		`  <ErrNo _="-2196" />`
		`  <ErrText _="path to key not found" />`
		`  <ErrorCause>`
		`  <ErrNo _="-2196" />`
		`  <ErrText _="path to key not found" />`
		`  <Class _="TXSTCPGetSetValueCmd" />`
		`  </ErrorCause>`
		`</Error>`

v	Fehlermeldung, lange Beschreibung	`<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>`

Tabelle 1-4: Fehlerrückgabewerte in Abhängigkeit des Parameters "ver"

2 Die Zustände des Gateway Lifecycles



Siehe [1]

2.1 Rollout (gatewayRollout)

2.1.1 Start Rollout

An die im Kapitel 7 mit CloudBaseUrl festgelegte URL werden vom Gateway die nachfolgenden Daten an den Broker gesendet. Wenn in der CloudConn Datenbank die Parameter `gateway_id` und `customer_id` vordefiniert sind, wird der Rollout übersprungen.

Topic

```
fp/gatewayRollout
```

Body

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

… oder in TiXML:

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

Der Broker überprüft, ob die Gateway Seriennummer bekannt ist und ob der Rollout-Prozess für dieses Gateway erwartet wird.

Bei dieser Kommunikation wird vom FP Gateway eine `request_id` festgelegt.

Aufbau der request_id beim Rollout:
Geräteseriennummer_x
Wobei x bei jeder Rolloutanfrage erhöht wird. Durch die Verwendung der Seriennummer wird die request_id systemweit eindeutig und nur durch ein Gerät verwendbar.

Die Seriennummer (`$tixi_gateway_serial_number`) des Gateways kann über folgenden Pfad abgefragt werden: `http://[Gateway IP]/System/Properties/SerialNo`

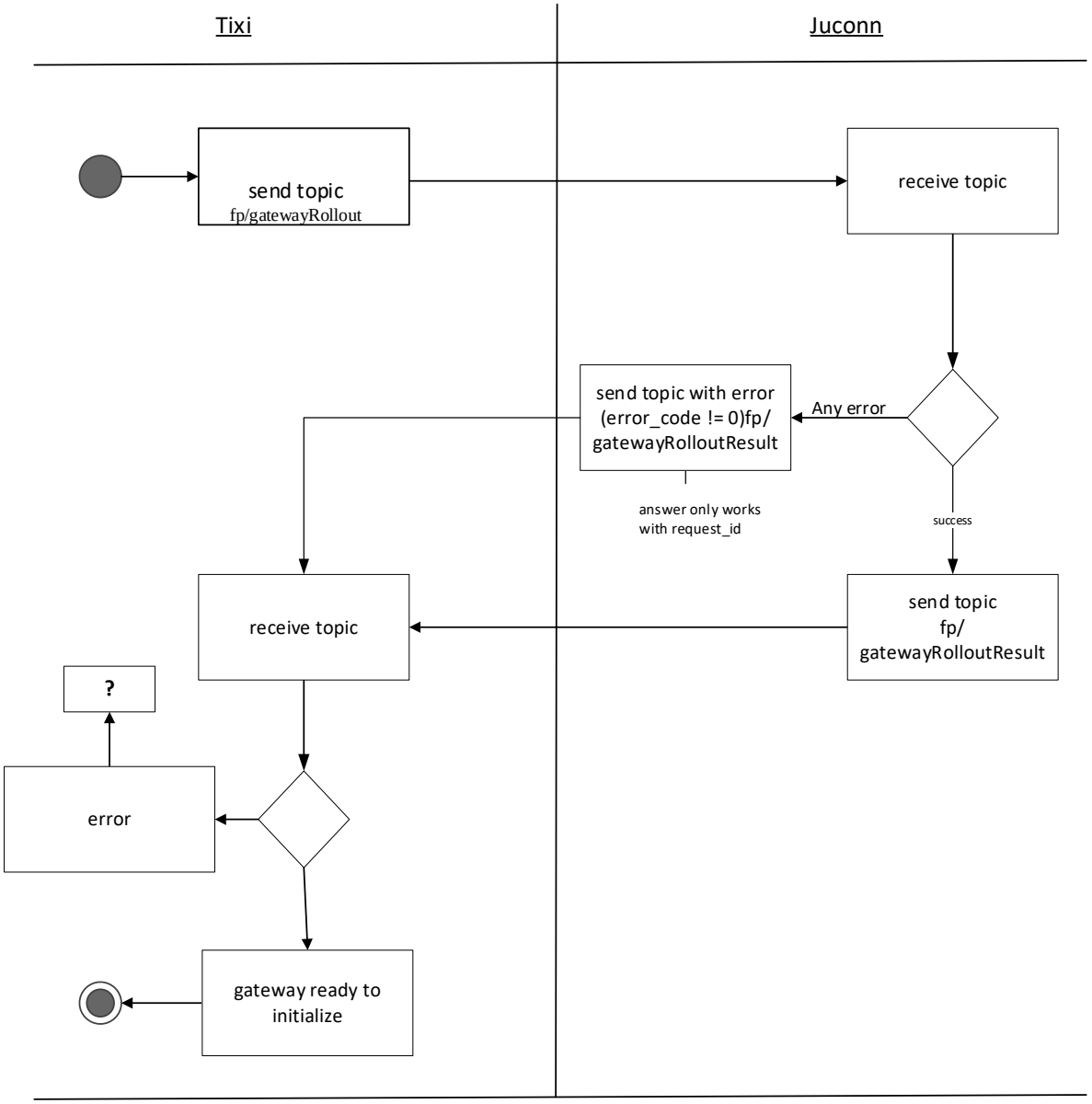
Der Produktname des Gateways (`$tixi_prodname`) kann über folgenden Pfad abgefragt werden: `http://[Gateway IP]/System/Properties/Hardware/Modules/Modem0`

Beispiel-Body für eine WandBox WT640 mit der Seriennummer `01234567`:

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

Die Geräteseriennummer ist 8-stellig und enthält ausschliesslich Ziffern.

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
receive topic
send topic
fp/
gatewayRolloutResult
?
error
gateway ready to
initialize

2.1.2 Wait for Rollout Result

Das Gateway wartet auf die Antwort des Brokers, der die Rollout-Anfrage wie folgt beantwortet:

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",
     //--------- Anfang optionaler Bereich --------
     "data_broker_details": {
        "url": "broker.juconn.io",
        "qos": 0
    },
     {
       "gateway_configuration": "gateway configuration details"
     }
     //---------- Ende optionaler Bereich ---------

}
```

... oder 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" />

     <!--------- Anfang optionaler Bereich -------->
     <data_broker_details>
        <url _="broker.juconn.io"/>
        <qos _="0"/>
     </data_broker_details>

     <gateway_configuration>
        gateway configuration details
     </gateway_configuration>
     <!---------- Ende optionaler Bereich --------->
}
```

$serial_number ist dabei die reale Seriennummer des Gateways (z.B. 04125378).

Wenn die `response_id` nicht mit der `request_id` aus dem vorangegangenen publish (fp/gatewayRollout) übereinstimmt, muss der Rollout Prozess erneut durchgeführt werden.
Die vom Broker übermittelten Credentials „gateway_id“ und „customer_id“ müssen permanent im Gateway gespeichert werden und werden später in allen Topics als Pfadangabe verwendet.
Bei einem Factory Reset bzw. einem `<Reset _="Secretfile"/>` müssen die Credentials wieder gelöscht werden.
Über die Variable Portal_Version wird die Software-Version des Brokers an das Gateway übermittelt. Dieser Wert soll vor allem den Support unterstützen. Anzeige der Portal_Version siehe Kapitel 7.5.

Über die optionale „gateway_configuration“ kann dem FP Gateway vom Broker aus eine Konfiguration übermittelt werden. Damit kann z.B. ein Kunde die gleiche Konfiguration auf all seine Gateways bekommen. Die „gateway_configuration“ enthält eine oder mehrere FP-Datenbanken im JSON oder TiXML Format.
Die „gateway_configuration” muss dann vom Gateway aktiviert werden. Dazu sollen die SetConfig-Befehle einzeln ausgeführt werden und das Ergebnis jeweils in das System-Logfile „SupportLog“ eingetragen werden.

Beispiel für eine 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}
        }
     }
  ]
 }
}
```

… oder 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>
```

Im nächsten Schritt „Gateway Initialisierung“ werden die am Gateway angeschlossenen Sensoren bzw. Aktoren dem Broker übermittelt.

2.2 Gateway Initialisierung (gatewayInit)

2.2.1 Start gatewayInit

Während der „Gateway Initialization“ sendet das Gateway die Definitionen aller Datenpunkte zum Portal, die später dort verarbeitet werden sollen. Dies dient dazu, das Datenmodell des Gateways mit dem Datenmodell des Portals abzugleichen. Die Gateway Initialization wird bei jedem Anmeldevorgang des Gateways am Portal durchlaufen, damit das Gateway-Datenmodell jederzeit konsistent mit dem Datenmodell des Portals ist.

Topic

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

Body

Prinzipieller Aufbau des Bodys:

```
{
     "request_id": "generated string",
     //--------- Anfang optionaler Bereich --------
     "virtual_devices": "virtual_devices_flag",
```

```
    "gateway_details": "TBD: gateway details",
    //--------- Anfang optionaler Bereich --------
    "controllers": [
      {
       controller 1 details
      },
      {
       controller 2 details
      },
      {
       controller n details
      }
   ]
}
```

Definition `virtual_devices`

Das optionale Flag“`virtual_devices`” legt fest, ob in der Cloud Datenpunkte zu virtuellen Devices (Controllern) zusammengefasst werden sollen. Die Definition des Flags erfolgt in der CloudConn-Datenbank über das Tag „`virtual_devices`“. Mögliche Werte sind:
0: Cloud soll keine virtuellen Devices anlegen (Default-Wert; Verhalten wie bisher)
1: Cloud soll virtuelle Devices anlegen (neues Verhalten)

Wenn das CloudConn-Flag „`virtual_devices`“ auf 1 steht, dann soll die Cloud-Business-Logik ausschließlich virtuelle Devices (Controller) anlegen. Die Zuordnung der Datenpunkte zu den virtuellen Devices erfolgt über das Tag „`meta_virtual_devices`“, welches für jeden Datenpunkt in den Realtime-Sektionen der CloudConn-Datenbank definiert sein muss.
Wenn das CloudConn-Flag „`virtual_devices`“ auf 1 steht, dann müssen alle Datenpunkte aller Realtime-Sektionen über das Tag „`meta_virtual_device`“ virtuellen Devices zugeordnet werden.
Der Name des virtuellen Devices kann frei definiert werden, sollte aber nur ASCII-Zeichen nutzen und darf keine Leerzeichen enthalten.

Controller Details

Die „controller details“ enthalten alle Datenpunkte des Gateways, die später zyklisch oder per Event zum Broker gesendet werden sollen (data model = device inventory). Dabei werden auch Metadaten zur genaueren Klassifizierung der Datenpunkte übertragen.

Ein Juconn-Controller ist ein Device in der Gateway-Welt.

Definition controller `name`

1. Variablen-Zugriffspfad > 1 Ebene
   Controller name = Pfadname bis zur vorletzten Hierarchie
   $Controller_ name_x bezeichnet den Inhalt der jeweiligen Variablen
   $Sensor_id_x bezeichnet den Inhalt der jeweiligen Variablen
   Slashes werden durch Unterstriche ersetzt

Beispiele (die Sensor Id ist in der Tabelle **fett** gedruckt):

FP-Variablenpfad (Get..)	Controller name	Kommentar
/GSM/**State**	GSM	GSM-Infos
/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-Modul, Bit-Zugriff
/Process/C540/QB/**P0**	Process_C540_QB	S1-Modul, Byte-Zugriff
/Process/C540/QW/**P0**	Process_C540_QW	S1-Modul, Word-Zugriff
/Process/C540/QD/**P0**	Process_C540_QD	S1-Modul, DWord-Zugriff
/Process/C03e/Counter/**P0**	Process_C03e_Counter	Erweiterungsmodule
/Process/MB/IO/I/**P0**	Process_MB_IO_I	Digitaler Eingang
/Process/MB/IO/Q/**P0**	Process_MB_IO_Q	Digitaler Ausgang

Tabelle 2-1: Variablen-Zugriffspfad > 1. Ebene

2. Variablen-Zugriffspfad = 1 Ebene
   Controller name = Get
   Slashes werden durch Unterstriche ersetzt

Beispiele (der Variablenname ist in der Tabelle **fett** gedruckt):

FP-Variablenpfad (Get..)	Controller name	Kommentar
/SerialNo	Get	Seriennummer
/GprsLinkState	Get	Linkstatus GPRS
/FreeFileSize	Get	Freier Flash-Speicher

Tabelle 2-2: Valriablen-Zugriffspfad = 1. Ebene

Definition der Controller Metadaten

Folgende Metadaten sind optional. Wie diese Daten in der TiXML-Konfiguration abgelegt werden, muss noch spezifiziert werden.

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

Definition Sensordaten

`external_sensor_id:` eindeutig innerhalb des Controllers

`name:` Name des Sensors; eindeutiger Name des Sensors; der Inhalt wird im Folgenden mit $Varname_x bezeichnet

-> ergibt sich aus dem XML-Namen des Datenpunktes in der CloudConn Datenbank

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

Im o.g. Beispiel ist $`Varname_1` der Name des Sensors.

Definition von Metadaten für Sensoren

Die Metadaten des Sensors werden in den Realtime-Zweigen (maximal 5 Zweige können definiert werden) der CloudConnector-Datenbank für jeden Datenpunkt definiert. Die Daten werden dort hinter dem Zugriffspfad der Variable als zusätzliche Tags definiert:

```
<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>
```

Alle in der CloudConn Datenbank definierten Tags müssen in den Juconn-Namen übersetzt werden, indem der Text „`meta_`“ im Tagnamen weggelassen wird. Aus `meta_type="temperature"` wird damit `type="temperature"`. Diese Juconn-Tags werden dann als Sensor-Metadaten an das Portal übermittelt. Tabelle 2-3 listet alle möglichen Metadaten auf.

Neu: Virtuelle Devices

Es ist möglich, beliebige Datenpunkte virtuellen Devices in der Cloud zuzuordnen. Diese Gruppierung erlaubt es dem Anwender, Daten aus verschiedenen Quellen zusammenzufassen und damit übersichtlich im Cloud-Dashboard darstellen zu können.
Dazu werden die Metadaten in den Realtime-Sektionen um das xml-Tag „`meta_virtual_device`“ erweitert, womit der Name des virtuellen Devices festgelegt wird und damit die Gruppierung der Daten erfolgt. Die Übertragung des Parameters erfolgt während der `gatewayInit`-Phase über den Metadaten-Parameter `virtual_device`.

Die über das xml-Tag "`meta_virtual_device`" definierten Gruppennamen werden in der Cloud als Namen für die virtuellen Devices (Controller-Name) verwendet.

Beispiel:

```
<Realtime1>
  <Varname_1  _="/Process/PV/MyProcessVar1" meta_type="humidity"
              meta_unit="%" meta_setpoint="30"
              meta_virtual_device="Heizung_1" />
  <Varname_2  _="/Process/MBus/Device0/Var0" meta_type="light"
              meta_unit="lm" meta_receive_signal="20"
              meta_virtual_device="Heizung_1" />
  <Varname_3  _="/Process/PV/MyProcessVar2" meta_type="temperature"
              meta_unit="K" meta_setpoint="5"
              meta_virtual_device="Heizung_2" />
  <Varname_4  _="/Process/MBus/Device1/Var0" meta_type="power"
              meta_unit="W" meta_receive_signal="10"
              meta_virtual_device="Heizung_2" />
  <Varname_5  _="/Process/Modbus/Device0/Var0" meta_type="voltage"
              meta_unit="V" meta_virtual_device="Heizung_2" />
</Realtime1>
```

Im Beispiel werden die Datenpunkte `Varname_1` und `Varname_2` dem virtuellen Device „`Heizung_1`“ zugeordnet. Die Datenpunkte `Varname_3`, `Varname_4` und `Varname_5` werden dem virtuellen Device „`Heizung_2`“ zugeordnet.

Auszug aus der `gatewayInit` Payload (hier werden der Einfachheit halber nur Sensors gezeigt) für die beiden ProcessVars (PV) aus dem oberen Beispiel:

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

```
      <set_point _="30" />
      <virtual_device _="Heizung_1"/>
    </$sensor_id_1>
    <$sensor_id_2>
      <name _="Varname_2"/>
      <type _="temperature"/>
      <unit _="K"/>
      <set_point _="5" />
      <virtual_device _="Heizung_2"/>
    </$sensor_id_2>
  </sensors>
```

Die Anzahl der einem virtuellen Device zugeordneten Datenpunkte ist nicht beschränkt.

Tabelle 2-3 zeigt die möglichen Metadaten.

Name in CloudConn DB	Erforderlich?	Juconn-Name	Inhalt
`meta_type`	ja	type	Sensortyp
`meta_unit`	nein	unit	Werte siehe Tabelle 2-4, Spalte „Base Unit“
`meta_setpoint`	nein	setpoint	Bezieht sich auf meta_type
`meta_virtual`	nein	virtual	`true` oder `false`
`meta_receive_signal`	nein	receive_signal	`REGULAR` oder `IRREGULAR` Definiert ob Sensordaten regelmäßig gesendet werden
`meta_signal_frequency`	nein	signal_frequency	in Sekunden Dieser Wert wird erwartet, wenn die Sensordaten regelmäßig gesendet werden sollen
`meta_custom_fields`	nein	custom_fields	
`meta_virtual_device`	nein	virtual_device	Gruppiert Datenpunkte zu virtuellen Devices

Tabelle 2-3: Übersicht der meta-Daten

Tabelle 2-4 zeigt die von Juconn unterstützten Sensortypen mit ihren Einheiten:

Sensortyp	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 *	beliebig	

Tabelle 2-4: Übersicht der Sensortypen

* Sensortyp „custom“: beliebiger Typ; keine Base Unit. Kann z.B. für CELLID etc. verwendet werden. Der Wert des Sensortyp „custom“ kann beliebig sein, also z.B. ein String oder eine Zahl in hexadezimaler Darstellung. Daher muss der Wert ohne Interpretation als String übertragen werden.

Bitte beachten: Sonderzeichen müssen in der TiXML-Darstellung über Entitäten dargestellt werden (z.B. das Grad-Zeichen ° muss als `&#xb0;` dargestellt werden).

Body Beispiel für gatewayInit (JSON):

```
{
     "request_id": "generated string",
     //--------- Anfang optionaler Bereich --------
     "virtual_devices": "1",
     "gateway_details" : "TBD any details",
     //---------- Ende optionaler Bereich ---------
     "controllers" : [
          {
               "name": " Controller 1 name",
               "external_controller_id" : "$controller_name_1",
     //--------- Anfang optionaler Bereich --------
               "custom_fields" : null,
               "timeout" : 360,
```

```
            //---------- Ende optionaler Bereich ---------
            "sensors" : [
                {
                    "external_sensor_id": "$sensor_id_1",
                    "name" : $Varname_1,
                        // z.B. $Varname_1 z.B. „Temp Sensor“
                    für Controller 1
                    "type" : "temperature",
                    //--------- Anfang optionaler Bereich --------
                    "unit" : "°C",
                    "set_point" : "",
                    "virtual" : false,
                    "virtual_device" : "Heizung_1",
                    "receive_signal" : "enum [REGULAR, IRREGULAR]",
                    "signal_frequency" : 300,
                    "custom_fields" : {
                        "plant": "Plant 1",
                        "loop" : "loop 1"
                    }
                //---------- Ende optionaler Bereich ---------
                },
                {
                    "external_sensor_id": “$sensor_id_2”,
                    "name" : $Varname_2,
                        // z.B. $Varname_2 z.B. “Humidity
                    Sensor” für Controller 1

                    "type" : "humidity",
                    //--------- Anfang optionaler Bereich --------
                    "unit" : "%",
                    "set_point" : "",
                    "virtual" : false,
                    "virtual_device" : "Heizung_2",
                    "receive_signal" : "enum [REGULAR, IRREGULAR]",
                    "signal_frequency" : 300,
                    "custom_fields" : null
                    //---------- Ende optionaler Bereich ---------
                }
            ]
        },
        {
            "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,
                        // z.B. $Varname_1 z.B. “Temp Sensor”
                    für Controller 2

                    "type" : "temperature",
                    //--------- Anfang optionaler Bereich --------
                    "unit" : "°C",
                    "set_point" : "",
                    "virtual" : false,
                    "virtual_device" : "Heizung_1",
                    "receive_signal" : "enum [REGULAR, IRREGULAR]",
                    "signal_frequency" : 300,
                    "custom_fields" : {
                        "plant": "Plant 1",
```

```
                                "loop" : "loop 1"
                            }
                            //---------- Ende optionaler Bereich ---------
                        },
                        {
                            "external_sensor_id": "$sensor_id_2",
                            "name" : $Varname_2,
                                // z.B. $Varname_2 z.B. “Humidity
                            Sensor” für Controller 2

                            "type" : "humidity",
                            //--------- Anfang optionaler Bereich --------
                            "unit" : "%",
                            "set_point" : "",
                            "virtual" : false,
                            "virtual_device" : "Heizung_2",
                            "receive_signal" : "enum [REGULAR, IRREGULAR]",
                            "signal_frequency" : 300,
                            "custom_fields" : null
                            //---------- Ende optionaler Bereich ---------
                        }
                    ]
                }
            ]
        }
```

… oder in TiXML:

```
{
    <?xml version="1.0" encoding="UTF-8" ?>
    <request_id _="generated string"/>
    <virtual_devices _="1" />
    <!-- ---------- Anfang optionaler Bereich --------- -->
    <gateway_details>TBD any details</gateway_details>
    <!-- ---------- Ende optionaler Bereich --------- -->
    <controllers>
      <$controller_name_1>
        <custom_fields/>
        <timeout _="360"/>
        <sensors>
          <$sensor_id_1>
            <name _=$Varname_1/>
            <!-- $Varname_1 z.B. “Temp Sensor”für controller_id1 -->
            <type _="temperature"/>
            <!--------- Anfang optionaler Bereich -------->
            <unit _="°C"/>
            <set_point />
            <virtual _="false"/>
            <virtual_device _="Heizung_1"/>
            <receive_signal _="enum [REGULAR, IRREGULAR] "/>
            <signal_frequency _="300"/>
            <custom_fields>
               <plant _="Plant 1"/>
               <loop _="loop 1"/>
            </custom_fields>
            <!---------- Ende optionaler Bereich ---------->
          </$sensor_id_1>
          <$sensor_id_2>
            <name _=$Varname_2/>
            <!-- $Varname_2 z.B. “Humidity Sensor”für controller_id1 -->
```

```
            <type _="humidity"/>
            <!--------- Anfang optionaler Bereich -------->
            <unit _="%"/>
            <set_point />
            <virtual _="false"/>
            <virtual_device _="Heizung_2"/>
            <receive_signal _="enum [REGULAR, IRREGULAR] "/>
            <signal_frequency _="300"/>
            <custom_fields/>
            <!---------- Ende optionaler Bereich ---------->
          </$sensor_id_2>
        </sensors>
      </$controller_name_1>
      <$controller_name_2>
       <custom_fields />
       <timeout _="360"/>
       <sensors>
          <$sensor_id_1>
            <name _=$Varname_1/>
            <!-- $Varname_1 z.B. “Temp Sensor” für controller_id2 -->

            <type _="temperature"/>
            <!--------- Anfang optionaler Bereich -------->
            <unit _="°C"/>
            <set_point />
            <virtual _="false"/>
            <virtual_device _="Heizung_1"/>
            <receive_signal _="enum [REGULAR, IRREGULAR] "/>
            <signal_frequency _="300"/>
            <custom_fields>
               <meta_plant _="Plant 1"/>
               <loop _="loop 1"/>
            </custom_fields>
            <!---------- Ende optionaler Bereich ---------->
        </$sensor_id_1>
        <$sensor_id_2>
          <name _=$Varname_2/>
           <!-- $Varname_2 z.B. “Humidity Sensor” für controller_id2 -->

          <type _="humidity"/>
          <!--------- Anfang optionaler Bereich -------->
          <unit _="%"/>
          <set_point />
          <virtual _="false"/>
          <virtual_device _="Heizung_2"/>
          <receive_signal _="enum [REGULAR, IRREGULAR] "/>
          <signal_frequency _="300"/>
          <custom_fields/>
          <!---------- Ende optionaler Bereich ---------->
        </$sensor_id_2>
       </sensors>
      </$controller_name_2>
    </controllers>
}
```

„controllers“ und „sensors“ müssen entsprechend dem realen Ausbau angepasst werden. Damit übernimmt der Broker die übermittelten Controller und Sensoren und ist bereit Daten zu empfangen. Controller_id1, controller_id2, sensor_id1 und sensor_id2 sind jeweils eindeutige Bezeichner innerhalb des aktuellen Zweiges.
Im Fehlerfall muss das FP Gateway den Rollout Prozess wiederholen.

2.2.2 Wait for Init Result

Die Initialisierung wird vom Broker abgeschlossen, in dem das Ergebnis an das Gateway über folgenden Topic zurückgeschickt wird:

Topic

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

Body

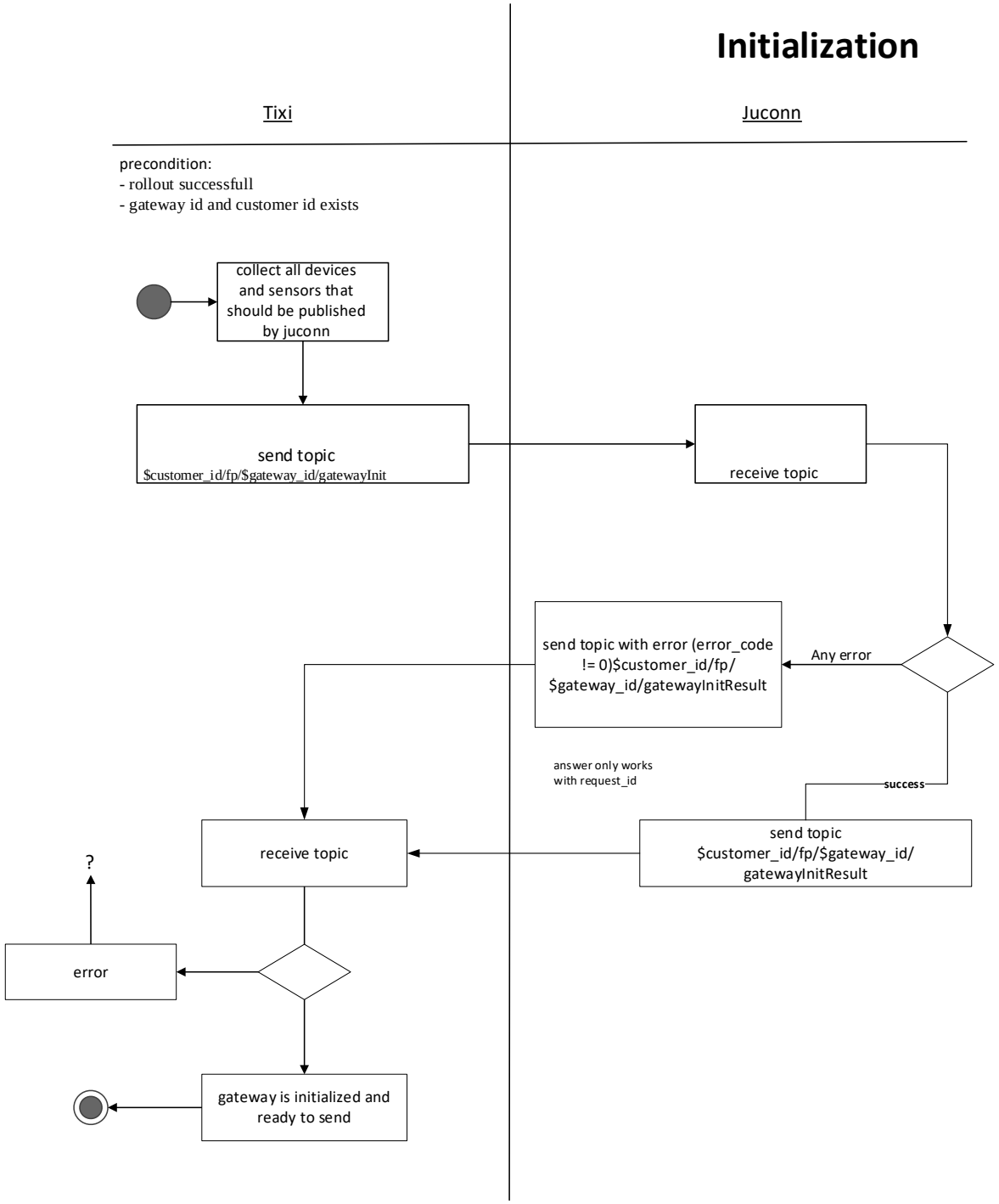
```
{
    "error_code": 0,
    "response_id": "the received request_id"
    "portal_version": "Portal_Version",
}
```

… oder in TiXML:

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

Über die Variable Portal_Version wird die Software-Version des Brokers an das Gateway übermittelt. Dieser Wert soll vor allem den Support unterstützen. Anzeige der Portal_Version siehe Kapitel 7.5.

Das Gateway ist nun bereit um Sensordaten an den Broker zu verschicken.


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

Beim Versenden der Daten, können entweder die Daten eines einzelnen Sensors, eines einzelnen Controllers oder Daten mehrere / aller Controller verschickt werden.

> Die message_id muss eine aufsteigende Dezimalzahl repräsentieren und wird vom Broker genutzt, um doppelt versandte Daten zu ignorieren.
> Die message_id kann beim Empfang von
> `$customer_id/fp/$gateway_id/gatewayInitResult`
> auf Null gesetzt werden.

2.3.1 Versenden der Daten eines Sensors

Mit diesem Topic wird der Wert eines Sensors übermittelt.
Der Versand von einzelnen Datenpunkten wird dann verwendet, wenn

a) In der Realtime-Sektion innerhalb der CloudConn Database nur ein Datenpunkt vorhanden ist
b) Bei Nutzung der Hysterese-Funktion aktuell nur ein Datenpunkt zu versenden ist

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
                }
           ]
        }
    ]

}
```

… oder 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>
}
```

Der Zeitstempel `timestamp` ist die Unix-Zeit (epoch). Zeitzone ist UTC, sofern nichts anderes in der CloudConn-Datenbank konfiguriert ist.

Die TiXML-Definition und Payload von „gps“-Sensoren ist in Kapitel 7.1 beschrieben. Der „`sensor_state`“ gibt Auskunft über den Variablentyp und teilweise über deren Status.
Siehe Tabelle 2-5.

Werte für sensor_state	Zustandsbeschreibung
0 oder 1	zeigt den DeviceState einer Busvariablen an 0 = Daten ungültig; 1 = Daten gültig
2	zeigt an, dass es sich um eine Process-Variable, eine Variable aus einem Erweiterungsmodul oder eine Variable mit Positionsdaten (GNSS / GPS -> Sensor Type = gps) handelt, für die es keinen DeviceState gibt

Tabelle 2-5: Kodierung des Sensorstatus

2.3.2 Verschicken der Daten eines Controllers

Mit diesem Topic werden alle Sensorwerte eines Controllers übermittelt.
Der Versand von einzelnen Datenpunkten wird dann verwendet, wenn

a) In der Realtime-Sektion innerhalb der CloudConn Database nur Datenpunkte eines Controllers vorhanden sind. Es müssen mindestens 2 Datenpunkte zu verschicken sein.
b) Bei Nutzung der Hysterese-Funktion aktuell nur Datenpunkte eines Controllers zu versenden sind und mindestens 2 Datenpunkte zu verschicken sind.

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
                }
          ]
        }
    ]
}
```

… oder 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>
}
```

Im oben dargestellten Beispiel ist id_3 ein Sensor vom Typ „gps“ (spezielle Darstellung).
Die TiXML-Definition und Payload von „gps“-Sensoren ist in Kapitel 7.1 beschrieben.

Der Zeitstempel `timestamp` ist die Unix-Zeit (epoch). Zeitzone ist UTC, sofern nichts anderes in der CloudConn-Datenbank konfiguriert ist.

2.3.3 Verschicken der Daten mehrerer Controller

Mit diesem Topic werden die Sensorwerte aller Controller an den Broker übermittelt.
Die Versandart „mehrerer Controller“ ist die Standard-Versandart und für alle Zwecke geeignet

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
                }
            ]
        }
    ]
}
```

Oder 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>
}
```

Der Zeitstempel `timestamp` ist die Unix-Zeit (epoch). Zeitzone ist UTC, sofern nichts anderes in der CloudConn-Datenbank konfiguriert ist.

Im oben dargestellten Beispiel ist id_3 ein Sensor vom Typ „gps“ (spezielle Darstellung).
Die TiXML-Definition und Payload von „gps“-Sensoren ist in Kapitel 7.1 beschrieben.

3 Sending Config Data

Konfigurationsparameter können über das nachfolgende Topic vom Broker zum FP Gateway gesendet werden. Es soll sowohl möglich sein, SetConfig-Befehle zu senden als auch Set-Kommandos.

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"
}
```

Oder 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>
}
```

Beispiel 1 für ein config_xml `SetConfig`-Statements:

```
    <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>
```

Beispiel 2 für ein config_xml `Set`-Statement:

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

Stimmt die serial_number nicht mit der im Gerät gespeicherten überein oder ist der enthaltene SetConfig Befehl fehlerhaft, gibt es einen entsprechenden Fehlercode in der anschließenden Antwort. Nur wenn die serial_number mit der im Gerät gespeicherten übereinstimmt, wird der SetConfig Befehl ausgeführt.

Die Konfiguration wird vom Gateway abgeschlossen mit:

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",
}
```

… oder 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"/>
}
```

Bzw.

```
{
    <?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>
}
```

Ein fehlerfrei ausgeführter <SetConfig> … </SetConfig> Auftrag wird mit <SetConfig/> beantwortet.

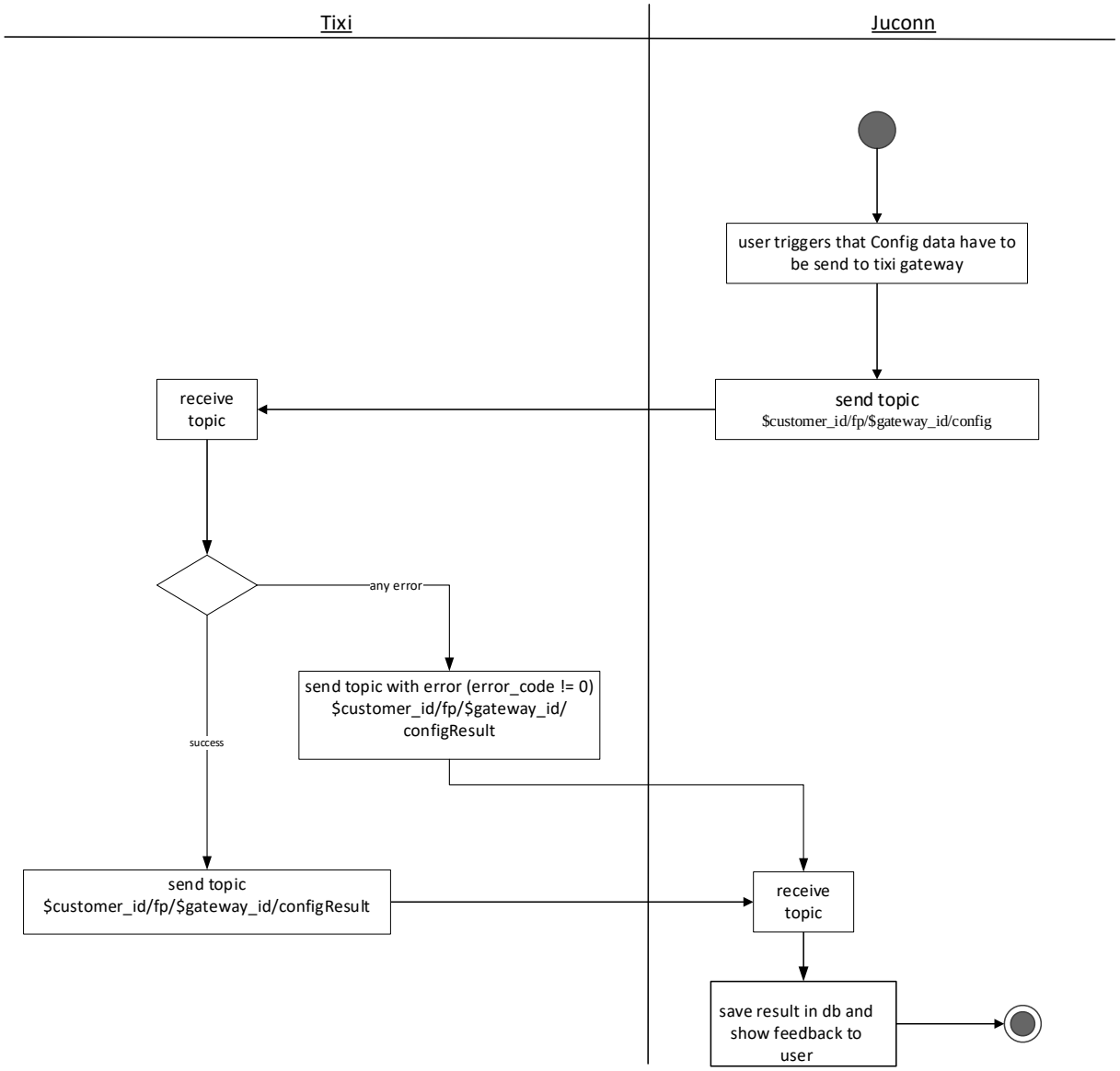
Wichtig:

Vor jedem SetConfig-Konfigurationsbefehl (also alle `SetConfig`-Befehle; nicht gültig für `Set`-Befehle!) sollte immer die Prozessbearbeitung intern mit folgendem Befehl gestoppt werden:

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

Das Ergebnis des oben genannten Befehls sollte abgewartet werden. Nur wenn der o.g. Stop-Befehl erfolgreich ausgeführt wurde, sollte anschließend ein `SetConfig`-Befehl ausgeführt werden.

Sending Config Data from Juconn to Tixi Gateway



Je nach SetConfig-Befehl kann es zwischen wenigen Sekunden (z.B. USER-Datenbank) und mehreren Stunden (M-Bus Scan-Befehl) dauern, bis das Gateway auf den Befehl antwortet.
Dem Broker ist überlassen, wie weiterverfahren werden soll.

4 Sending Command to FP Gateway

Über einen Rückkanal kann der MQTT-Broker beliebige TiXML-Befehle an das FP-Gateway senden, um so zum Beispiel eine neue Konfiguration in das Gerät zu speichern, das Gerät neu zu starten oder einen Ausgang zu schalten. Der MQTT-Broker kann sämtliche TiXML-Befehle an das Gerät schicken, wie es auch mit der Konfigurationssoftware TICO möglich ist.
Der Client abonniert folgenden Topic beim Broker:

Topic

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

Body

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

… oder in TiXML:

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

Beispiel:

```
{
    "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" }
             }
          }
       }
    ]
   }
}
```

Oder 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>
   }
```

Das Kommando wird lediglich bei Übereinstimmung mit der serial_number des Gateways ausgeführt.

Possible Errors

code 500:
invalid command

Wichtig:
Vor jedem SetConfig-Konfigurationsbefehl (also alle `SetConfig`-Befehle; nicht gültig für alle anderen Befehle wie z.B. `Set`-Befehle, `DoOn` etc.!) sollte immer die Prozessbearbeitung intern mit folgendem Befehl gestoppt werden:

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

Das Ergebnis des oben genannten Befehls sollte abgewartet werden. Nur wenn der o.g. Stop-Befehl erfolgreich ausgeführt wurde, sollte anschließend ein `SetConfig`-Befehl ausgeführt werden.

Der Client schickt das Ergebnis der Befehlsbearbeitung über einen publish-Topic an den Broker zurück:

Topic

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

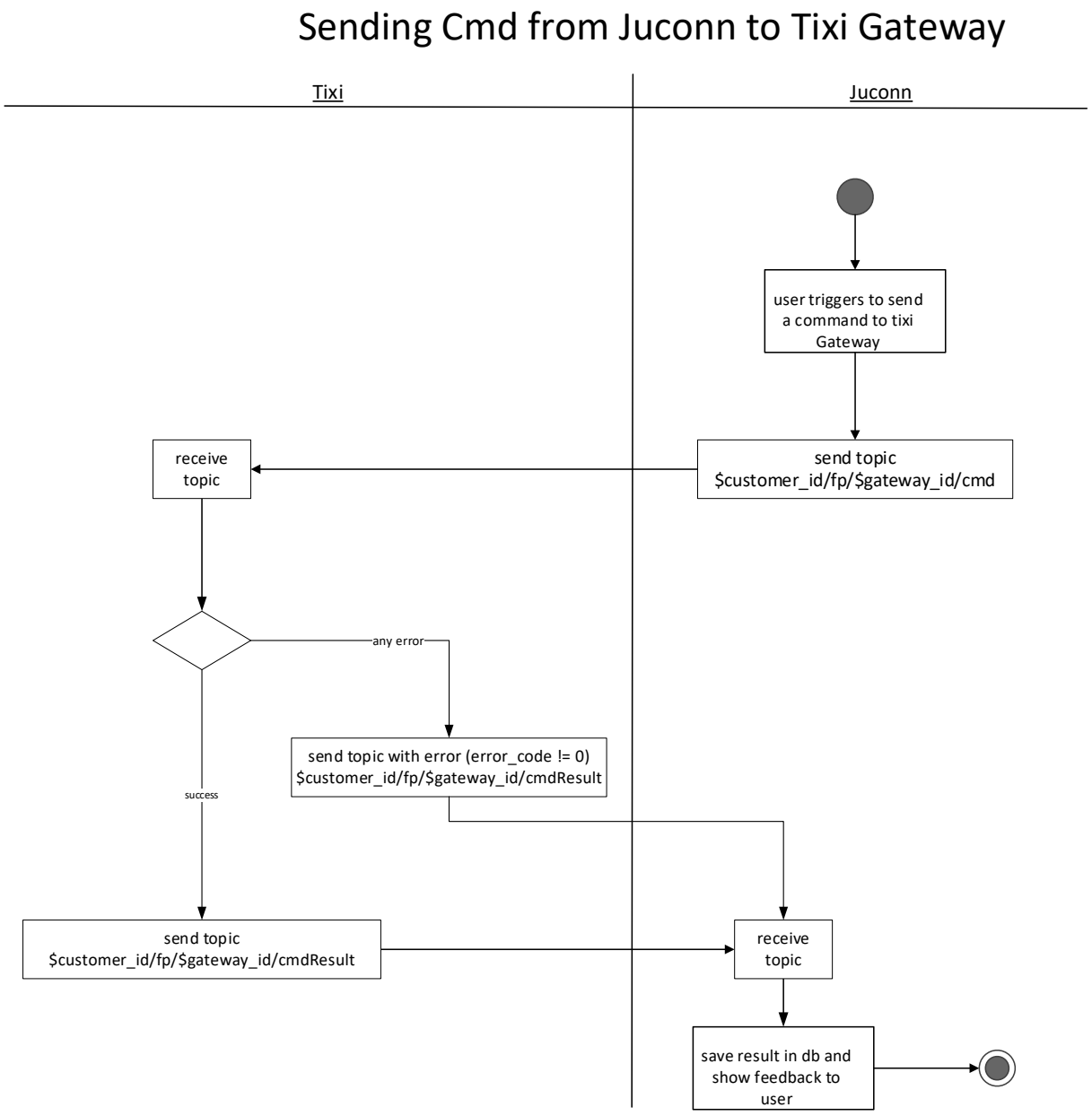
Body

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

… oder 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

Über einen State Request kann der Broker Gateway-spezifische Statusinformationen vom Gateway abfragen, z.B.

- Mobilfunkdaten (IMEI, IMSI, Provider, Empfangsfeldstärke, Dienst, …)
  Diese Daten sind nur dann verfügbar, wenn das Gateway über ein Mobilfunkmodul verfügt.
- Hardware-Infos (Speichergröße Flash, RAM, HW-Revision, MAC-Adresse)
- Verfügbare I/Os (Onboard-Schnittstellen, Erweiterungsmodule etc.)
- System-Infos (CPU-Load, Board-Temperatur, Ladezustand des Stützakkus, WiFi-Zustand)
- Zeit-Infos (Zeit, letzte Ausschaltzeit, letzte Einschaltzeit, Zykluszeiten)
- Software-Infos (Version von Uboot/Linux/Linux-App/Konfiguration)

Über folgenden Topic werden alle o.g. Statusinfos gesammelt an den Broker zurückgeschickt.

Topic

$customer_id/fp/$juconn_gateway_id/state

Body

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

… oder in TiXML:

```
{
    <?xml version="1.0" encoding="UTF-8" ?>
    <request_id _=”generated string“/>
    <serial_number _=$external_gateway_serial_number />
    <!-- optional path. Es soll nur der Pfad "Ethernet" gesendet werden -->
    <path _="Ethernet" />
}
```

Das FP Gateway beantwortet diesen Request mit:

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"
    }
}
```

Oder 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>
}
```

Der `gateway_state` liefert die Antwort auf das TiXML-Kommando `<Get ver="y" />`.
Wenn der Parameter path mitgeschickt wurde, dann soll das Gateway diesen Pfad im Get-Kommando berücksichtigen: `<Get _="path" ver="y" />`
Beispiel: `path _="Ethernet": <Get _="Ethernet/" ver="y" />`

Beispiel:

```
{
"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" }
    }
  }
}
}
}
```


... oder 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

Normalerweise wird die Verbindung zum Broker über das aktive LAN-Interface aufgebaut. Über die TiXML-Datenbank `ISP/ISP/OUT` kann die MQTT-Verbindung auch über eine Mobilfunkverbindung oder einen VPN-Tunnel aufgebaut werden:

```
[<SetConfig _="ISP/ISP" ver="y">
    <!-- Kommunikationsinterface fuer Dienste definieren -->
    <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>]
```

Die folgenden Optionen sind verwendbar für den MQTT-Client:

```
Ethernet
MODEM
VPN
```

7 TiXML-Konfiguration (CloudConn)

Der MQTT-Client wird über eine TiXML-Datenbank konfiguriert.

```
[<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_...]/>
        <!-- spezieller Datentyp: GNSS
           Wert des Dummy-Attributs ist Latitude, lng-Wert ist Longitude
        -->
        <GNSS _="/GNSS/Latitude" lng="/GNSS/Longitude" meta_type="gps" />
      </Realtime1>

     </CloudConn>

</SetConfig>]
```

7.1 Nutzung des Sensortyps „gps“

Juconn kennt den Sensortyp „gps“, bei dem lng (Longitude) und lat (Latitude) übertragen werden. Da FP Gateways keinen Datentyp mit mehr als einem Wert unterstützen , wird der Typ „gps“ speziell in der CloudConn Database kodiert und dann bei der Übertragung der Payload aus 2 Werten zusammengesetzt.

Sobald der meta_type="gps" auftaucht, muss ein zweiter Wert (lat="") definiert sein. Der State wird statisch auf "2" gesetzt (wie bei ProcessVars). Dieses Wertepaar muss dann gemeinsam als Payload übertragen werden.

Fehlerbehandlung:
Wenn die gps-Werte nicht auflösbar sind (weil nicht vorhanden), dann sollen leere Werte übermittelt werden.

Payload Beispiel 1 (Werte existieren):

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

Payload Beispiel 2 (Werte existieren NICHT):

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

7.2 Hysterese

Für den Timer 1 (`RTsendTime1`) ist optional für jeden Datenpunkt einstellbar, wann dieser abhängig von einer Hystereseeinstellung übertragen werden soll.

Beispiel: Datenpunkt „Temperatur“
Startwert = 20,5K
Hysterese absolut = 0,5

Ergebnis: Wenn der Wert des Datenpunktes auf 20 oder 21 wechselt, wird dieser übertragen.

Festlegungen:

- Nach dem Neustart des Gerätes werden alle Werte einmalig übertragen.
- Wenn sich der zuletzt übertragene Wert um den Hysterese-Wert ändert, wird der Wert wieder einmalig übertragen.

Die benötigten Parameter:

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

Der Wert `-1` bedeutet, dass Daten nicht mehr zyklisch gesendet werden, sondern anhand der Hysterese-Parametrierung.

`<UpdateRate1 _="Pollzeit" />`

Pollzeit spezifiziert, wie häufig die Werte aus dem Process-Zweig abgefragt werden sollen. Die Pollzeit wird in Sekunden angegeben. Wertebereich: 1 .. 3600

Achtung:
Die Hysteresefunktion ist derzeit nur für den ersten Timer verfügbar (`RTsendTime1 und UpdateRate1`). Die Hysterese ist nur für numerische Werte nutzbar.

Für jeden Realtime-Wert wird in der Konfiguration die Hysterese als Fließkommazahl festgelegt.
Beispiel:

```
hysteresis="1.5"
```

Definition:

- Komma / Dezimalpunkt wird mit "." dargestellt
- Max. 2 Nachkommastellen
- keine negativen Zahlen

Beispiel:

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

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

Im oben gezeigten Beispiel wird die Hysterese der einzelnen Datenpunkte festgelegt.
Wenn sich der Datenpunkt `Voltage_L1_N` um +/- 2.0 zum zuvor übertragenen Wert ändert, wird dieser Wert übertragen. Wenn sich der Datenpunkt `CPULoad` um +/- 8.5 zum zuvor übertragenen Wert ändert, wird dieser Wert übertragen usw.

7.3 Informationen über Realtime-Sektionen

In bis zu 5 Realtime Sektionen kann festgelegt werden, welche Datenpunkte zu übertragen sind.

Jeder Realtime Sektion muss ihr Zyklus vorangestellt werden.

z.B.
RTSendTime1, …, RTSendTime5.
Dazu gehören dann die Daten aus den entsprechenden Sektionen <RealTime1>..</RealTime1> usw.

Beispiel (siehe hierzu auch 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>
```

Datenpunkte können per EventHandler oder auch per Kommando verschickt werden:

Kommando (TICO, …):

```
    <CloudSendRealtimeData [_="CloudConn[,Realtime_Specifier"]] />
```

CloudConn (optional) = Name des CloudConnectors. Werte: CloudConn, CloudConn2, CloudConn3. Default: CloudConn
Realtime_Specifier (optional) = Name der Realtime-Sektion, z.B. Realtime1. Default=Realtime1
Wird der optionale Parameter weggelassen, werden alle definierten Sektionen versendet.

Beispiel 1: Realtime1-Sektionen von CloudConn versenden
```
<CloudSendRealtimeData />
```

Beispiel 2: Realtime-Sektion "Realtime2" von CloudConn2 versenden
```
<CloudSendRealtimeData _="CloudConn2,Realtime2" />
```

Beispiel 3: Realtime-Sektion "Realtime3" von CloudConn versenden
```
<CloudSendRealtimeData _="CloudConn,Realtime3" />
```

7.4 Wichtige Hinweise zur Nutzung des MQTT-Clients

Bitte beachten Sie folgende Hinweise:

- Die fett gedruckten Konfigurationsoptionen müssen konfiguriert werden (mandatory).
- Es muss mindestens eine CloudBaseUrl konfiguriert werden.
- Der Pfad für unverschlüsselte Urls lautet: **`tcp://URL-or_IP_of_MQTT_broker:1883`**
- Der Pfad für verschlüsselte Urls lautet: **`ssl://URL-or_IP_of_MQTT_broker:8883`**
- Es kann genau 1 CloudBaseUrl konfiguriert werden
- CloudConnStart und CloudConnType müssen wie oben beschrieben konfiguriert werden.
- Die Daten im Zweig Realtime1 können vom Anwender frei konfiguriert werden.

Nach dem Einspielen der MQTT-Client-Konfiguration sollte das FP Gateway neu gestartet werden.

7.5 Statusanzeige der MQTT-Verbindung

Statusanzeige im Process-Zweig
Den Verbindungstatus des MQTT-Client kann man im Process-Zweig überwachen:
```
[<Get _="/Process/CloudConn/" ver="y" />]
```

Ergebnis (Beispiel):
```
<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` zeigt die Art der Verbindung (unverschlüsselt = mqtt/verschlüsselt = mqtt+ssl), die URL / IP-Adresse des gerade aktiven Brokers und den MQTT-Port an (z.B. 1883 oder 8883).
`ConnectionState` = 0 bedeutet: keine Verbindung
`ConnectionState` = 1 Verbindung ist aufgebaut
`ConnectionStateMsg` = (registering, not connected, connecting, connected)

`LastTimeStamp` = (Zeitstempel der letzten Datenübertragung in Unix Time)
`ChangeToggle` = (wechselt jedesmal zwischen 0 und 1, sobald sich der Zeitstempel geändert hat)
`BrokerVersion` = (Version des Brokers, siehe Kapitel 2.1.2)

Optische Signalisierung über Signal-LED

Die „Signal“-LED signalisiert den Zustand der MQTT-Verbindung.
Aus = keine MQTT-Verbindung aktiv
Rot blinkend = MQTT-Verbindung wird aufgebaut
Grün leuchtend = MQTT-Verbindung ist aufgebaut

8 Portal Fehlercodes

Die aktuell bekannten Portal Fehlermeldungen sind vom State des Gateways abhängig und sind in der Tabelle 8-1 aufgeführt.

Im State	Wert	Bedeutung
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

Tabelle 8-1: Juconn Fehlercodes

9 Glossar

Ein begleitendes Glossar befindet sich: [2]

CloudConnector	ist ein Protokoll um FP Gateways mit Cloud Services zu verbinden
Gateway	Das Wort Gateway ['geɪtweɪ] (englisch für Ausfahrt und Einfahrt, wörtlich Torweg) bezeichnet in der Informatik eine Komponente (Hard- und/oder Software), welche zwischen zwei Systemen eine Verbindung herstellt
JSON	Java Script Object Notation
MQTT	Message Queuing Telemetry Transport (MQTT) ist ein offenes Nachrichtenprotokoll für die Machine-to-Machine-Kommunikation
TiXML	Simple XML Control Protocol, benutzt Extensible Mark-up Language (XML) XML-kompatible, vereinfachte Darstellung der XML-Daten (spart Zeichen ein)
URL	Uniform Resource Locator

10 Literaturverzeichnis

[1] Francotyp Postalia, Ralf Müller, „GatewayBrokerLifeCycle.pdf“.
[2] Juconn, Julian Dawo, „Tixi_WordingDefinition.pdf“.
[3] Juconn, Julian Dawo, „JUC_TixiIngeration_v4.pdf“.
[4] SPS-Programmierhandbuch: „510058920100_XX_FP-SPS-TiXML-Handbuch_DE.pdf“.
[5] Juconn, Julian Dawo, „Juconn_MqttMessages171123.pdf“.
[6] TiXML-Programmierhandbuch (EN): „510058920001_XX_FP-TiXML-Reference_EN“.

xx = Revisionsnummer des Dokuments, beginnend bei 00
Für weitere Informationen und zum Download der Dokumente siehe www.inovolabs.com.

11 Historie

Version	Änderung
1.3.1	Redaktionelle Bearbeitung im Kontext der ersten englischen Ausgabe
1.3.0	Neue Metadaten für Datenpunkte (virtual_devices), Textkorrekturen
1.2.9	Berichtigungen in Kapitel 7.2
1.2.8	Erweiterungen und Korrekturen in Kapitel 7.5
1.2.7	- neuer Dokumentname; neues Deckblatt - Fixes in Kapitel 5:Antwort auf Status Request über response_id, nicht request_id an weiterer Stelle Abschnitt 7.2 neu eingefügt Abschnitt 7.3 neu eingefügt
1.2.6	- Fixes in Kapitel 5:Antwort auf Status Request über response_id, nicht request_id
1.2.5	- Neuer Sensortyp „custom“; siehe Tabelle 2-4 - aktualisierte Beschreibung in Kapitel 3 - aktualisierte Beschreibung in Kapitel 4
1.2.4	- 2.1.1 Korrekturen im Rollout-Prozess (Seriennummer-Beispiel korrigiert)
1.2.3	- 2.1.2 Rollout-Prozess überarbeitet (Sicherheitslücke geschlossen) - Umformatierungen
1.2.2	- Tabelle 1-1, Tabelle 1-2 korrigiert - 2.1.1 Start Rollout: Request_id für Rollout korrigiert - 2.2.1 Start gatewayInit korrigiert - 2.3 Running korrigiert - 3 Sending Config Data Run/Stop Beispiel korrigiert - 8 Portal Fehlercodes eingefügt - 11 Historie eingefügt Achtung: die JSON Beispiele sind noch nicht final aktualisiert/korrigiert