

EWIO₂ Remote Control API

Inhaltsverzeichnis

1	Einleitung	3
2	Prinzipieller Aufbau einer Kommunikation.....	4
3	Übersicht Funktionen.....	5
3.1	Verwaltungsfunktionen.....	5
3.1.1	Beantragung Session-ID.....	5
3.1.2	Login	5
3.1.3	Logout	5
3.1.4	Reboot	6
3.2	Allgemeine Funktionen.....	7
3.2.1	File laden (Textfile - .json)	7
3.2.2	File speichern (Textfile - .json).....	7
3.2.3	Verzeichnisinhalt anfordern	8
3.2.4	Firmware aktualisieren.....	9
3.3	Geräte-Konfiguration laden / speichern.....	10
3.3.1	Geräte-Konfiguration laden.....	10
3.3.2	Geräte-Konfiguration speichern.....	10
3.4	Funktionen zu Applikationen	14
3.4.1	Applikationsliste laden	14
3.4.2	Applikation laden.....	14
3.5	M-Bus- Funktionen.....	15
3.5.1	Zähler-Liste laden	15
3.5.2	Zählerkonfiguration löschen.....	15
3.5.3	Zähler-Konfiguration speichern.....	16
3.5.4	Zähler-Konfiguration laden	16
3.5.5	Datenpunktliste laden	18
3.5.6	Datenpunkt-Konfiguration speichern	19
3.5.7	Datenpunkt-Konfiguration laden	19
3.5.8	M-Bus-Parameter für Dokumentation laden	21
3.5.9	M-Bus-Suche	22
3.5.10	M-Bus-Suchergebnisse – Zählerliste laden.....	24
3.5.11	M-Bus-Suchergebnisse – Datenpunktliste laden.....	24
3.6	Modbus – Funktionen.....	26
3.6.1	Zählertyp - Liste laden	26
3.6.2	Erweiterte Zählertyp - Liste laden	27
3.6.3	Zähler-Liste laden	27
3.6.4	Zählerkonfiguration löschen.....	28
3.6.5	Zähler-Konfiguration speichern.....	29
3.6.6	Zähler-Konfiguration laden	29
3.6.6	Datenpunktliste laden	31
3.6.7	Datenpunkt-Konfiguration speichern	32
3.6.8	Datenpunkt-Konfiguration laden	32
3.6.9	Modbus-Parameter für Dokumentation laden.....	33
3.7	Systemzähler – Funktionen.....	35
3.7.1	Zähler-Liste laden	35

3.7.2	Zählerkonfiguration löschen.....	35
3.7.3	Zähler-Konfiguration speichern.....	36
3.7.4	Zähler-Konfiguration laden	36
3.7.5	Datenpunktliste laden	38
3.7.6	Datenpunkt-Konfiguration speichern	39
3.7.7	Datenpunkt-Konfiguration laden	39
3.7.8	Systemzähler-Parameter für Dokumentation laden.....	40
3.8	Allgemeine Zählerfunktionen.....	41
3.8.1	Live-Check.....	41
3.9	SQL- Funktionen, allgemein.....	41
3.9.1	Eine SQL-Anweisung senden.....	42
3.9.2	Das Resultat einer SQL-Anweisung erhalten	42
3.10	SQL- Funktionen, speziell.....	43
3.10.1	Messwerte aus Datenbank laden	43
3.10.2	Messwerte in Datenbank speichern.....	43
4	Aufbau der Datenbank	45

1 Einleitung

Die Kommunikation zwischen EWIO₂ und Web-Interface erfolgt über HTTP auf TCP/IP. Die Applikationsschicht, welche mit HTTP übertragen wird, kann aber auch von anderen Anwendungen als einem Web-Browser bedient werden.

Prinzipiell kann somit alle Kommunikation, die vom Web-Interface erfolgt, auch durch Anwendungen, welche die API benutzen, ausgeführt werden.

Darüber hinaus stehen allgemeine Funktionen (wie für SQL-Anweisungen) zur Verfügung, mit denen zusätzliche Funktionalität ermöglicht wird.

2 Prinzipieller Aufbau einer Kommunikation

Jede Kommunikation wird mittels einer TAN geschützt. Deshalb ist es notwendig, vor dem eigentlichen Senden oder Empfangen von Nutzdaten eine Login-Prozedur durchzuführen. Innerhalb dieser können beliebig viele Kommunikationsschritte (Funktionen) durchgeführt werden.

Kommunikationsschritte: ID beantragen

Login

Funktion 1

Funktion 2

..

Logout

3 Übersicht Funktionen

In der Beschreibung der Funktionen wird der EWIO₂ als Server und der Browser oder Nutzer der API als Client bezeichnet.

3.1 Verwaltungsfunktionen

3.1.1 Beantragung Session-ID

Der Server generiert eine Zufalls-ID, die dem Client übermittelt wird.

Methode: GET
URL: <server-ip>/cgi-bin/getParamFromServer.cgi
Parameter:
username=Administrator&password=<pw-md5-hash>&type=session_id&module=login
Rückgabe: <Session-ID>

3.1.2 Login

Der Client sendet den MD5-Hash der Verkettung von Passwort und Session-ID und Passwort (= tan-md5-hash) an den Server. Dieser gibt bei korrektem tan-md5-hash ein LOGIN OK, andernfalls ein LOGIN ERROR zurück.

Methode: GET
URL: <server-ip>/cgi-bin/getParamFromServer.cgi
Parameter:
username=Administrator&password=<tan-md5-hash>&type=login&module=login
Rückgabe: LOGIN OK / LOGIN ERROR

3.1.3 Logout

Soll die Kommunikation von Seiten des Client beendet werden, sendet dieser ein Logout an den Server. Andernfalls wird nach 30 Minuten ohne Kommunikation das Logout vom Server automatisch veranlasst.

Methode: GET
URL: <server-ip>/cgi-bin/getParamFromServer.cgi
Parameter:
username=Administrator&password=<tan-md5-hash>&type=logout&module=login
Rückgabe: LOGOUT OK

3.1.4 Reboot

Soll die Kommunikation von Seiten des Client mit einem Neustart des EWIO₂ beendet werden, sendet dieser ein Reboot an den Server. Im Anschluss an die Übertragung der Rückgabe startet sich das EWIO₂ neu. Nach dem Neustart muss eine neue Session-ID beantragt und der Login-Vorgang wiederholt werden.

Methode: GET

URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:

username=Administrator&password= <tan-md5-hash>&type=reboot&module=login

Rückgabe: REBOOT OK

3.2 Allgemeine Funktionen

3.2.1 File laden (Textfile - .json)

Der Client fordert ein auf dem EWIO₂ vorliegendes Textfile an.

Methode: GET
URL: <server-ip>/cgi-bin/getTxtFileFromServer.cgi

Parameter:
username=Administrator&password=<tan-md5-hash>&type=fileload&module=<Quelladresse der Datei>

Rückgabe: <Dateiinhalt>

3.2.2 File speichern (Textfile - .json)

Der Client sendet an den Server ein Textfile.

Methode: POST
URL: <server-ip>/cgi-bin/setTxtFileToServer.cgi

POST-Parameter:

Host:<server-ip>

Content-Type:application/x-www-form-urlencoded

Content-Length:<length>

Parameter:
username=Administrator&password=<tan-md5-hash>&type=fileload&
module=<Zieladresse der Datei>&data=<Dateiinhalt>

Rückgabe: -

3.2.3 Verzeichnisinhalt anfordern

Der Client fordert den Inhalt des unter "modul" angegebenen Verzeichnisses an. Ergebnis ist der Verzeichnisinhalt im JSON-Format.

Methode: GET

URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:

username=Administrator&password= <tan-md5-hash>&type=dir_download&module= <path of dir>

Rückgabe: <JSON-Struktur>

```
{
  "files":
  [
    "counter.conf",
    "ewio2server.ini",
    "IoDriver.conf"
  ]
}
```

Bsp. für JSON-Struktur eines Verzeichnisinhalts, hier: Verzeichnis /var/opt/etc

3.2.4 Firmware aktualisieren

Der Client sendet eine Firmware-Datei an den Server, der damit die Gerätefirmware aktualisiert. Nach diesem Aufruf wird die Kommunikation mit dem Client durch den Server automatisch beendet (so als ob die Funktion "Logout" aufgerufen worden wäre). Das gilt auch, wenn der Aktualisierungsvorgang aus irgendeinem Grund fehlschlägt. Zur Fortsetzung der Kommunikation muss eine neue Session-ID beantragt werden.

Methode: POST

URL: <server-ip>/cgi-bin/setBinFileToServer.cgi

POST-Parameter:

Host:<server-ip>

Content-Type:multipart/form-data; boundary=...

Content-Length:<length>

Parameter (jeweils als separate Multipart-Formularfelder, durch den boundary-String getrennt):

username=Administrator

password=<tan-md5-hash>

type=firmware_update

module=<Name der Firmware-Datei> (optional)

data=<Dateiinhalt> (mit Content-Type application/octet-stream)

Rückgabe: <Text>

Die Rückgabe zeigt den Erfolg des Aktualisierungsvorgangs an. Im Erfolgsfall lautet der Text "OK". In diesem Fall wird das Gerät automatisch neu gestartet, wodurch die Firmware-Aktualisierung abgeschlossen wird. Jeder andere Text zeigt ein Fehlschlagen des Aktualisierungsvorgangs an. In diesem Fall wird das Gerät nicht neu gestartet und der Aktualisierungsversuch kann wiederholt werden (nach Beantragung einer neuen Session-ID). Ein fehlgeschlagener Aktualisierungsversuch beschädigt die auf dem Gerät laufende Firmware nicht.

Folgende Fehlermeldungen können zurückgegeben werden:

Fehlermeldung	Beschreibung
F_LEVEL	Unzureichende Zugriffsrechte. Nur der Nutzer Administrator hat die Berechtigung zur Firmware-Aktualisierung.
FW_INVSIG	Die Signatur der Firmware-Datei ist ungültig. Wahrscheinlich wurde die Datei beschädigt.
FW_UNTAR	Beim Extrahieren des Inhalts der Firmware-Datei ist ein Fehler aufgetreten.
FW_DOWNGRADE	Die in der Firmware-Datei enthaltene Firmware ist älter als die auf dem Gerät laufende Firmware.
FW_ERR	Während des Aktualisierungsvorgangs ist ein unerwarteter Fehler aufgetreten.

3.3 Geräte-Konfiguration laden / speichern

Die Geräte-Konfiguration ist in mehrere Themen aufgeteilt, die jeweils in einem Konfigurationsdatensatz zusammengefasst sind. Je nach Thema liegt der Konfigurationsdatensatz in einem Textformat oder im JSON-Format vor.

Zu folgenden Themen können Konfigurationsdatensätze geladen und gespeichert werden:

Thema	Modul-Parameter	Daten-format	Laden / Speichern	Beschreibung
Gerät	device	JSON	nur Laden	Gerätemodell und Seriennummer
Version	version	JSON	nur Laden	API-Version und Softwareversion
Datum und Zeit	datetime	JSON	Laden und Speichern	Konfiguration von Datum, Zeit, Zeitzone und Timeserver
Geräte-konfiguration	devicebase	JSON	nur Laden	Gerätemodell, Seriennummer, MAC-Adresse, Softwareversion und Auslastung des Flash-Speichers
Speicher	memory	JSON	nur Laden	Auslastung von Flash-Speicher und SD-Karte
I/O-Anschlüsse	io_driver	Text	Laden und Speichern	Konfiguration der I/O-Anschlüsse des EWIO ₂

3.3.1 Geräte-Konfiguration laden

Der Client fordert Konfigurationsdaten zu einem bestimmten Thema an. Das Thema wird als Parameter "modul" angegeben.

(Bsp.: Thema Datum und Zeit → modul=datetime)

Methode: GET

URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:

username=Administrator&password=<tan-md5-hash>&type=configuration&module=<Thema>

Rückgabe: <Konfigurationsdatensatz>

3.3.2 Geräte-Konfiguration speichern

Der Client sendet an den EWIO₂ Konfigurationsdaten zu einem Thema, die dort abgespeichert werden soll. Das Thema wird als Parameter "modul" angegeben.

(Bsp.: Thema Datum und Zeit → modul=datetime)

Die Konfiguration muss im Datenformat des Themas vorliegen. Bei Konfigurationsdatensätzen im JSON-Format nicht angegebene Daten werden unverändert beibehalten. Bei Konfigurationsdatensätzen im Textformat wird die gespeicherte Konfiguration als Ganzes wirksam (nicht angegebene Daten führen zu Default-Werten).

Methode: POST



URL: <server-ip>/cgi-bin/setParamToServer.cgi

POST-Parameter:

Host: <server-ip>

Content-Type: application/x-www-form-urlencoded

Content-Length: <length>

Parameter:

username=Administrator&password=<tan-md5-hash>&type=configuration&
module=<Thema>&data=<Konfigurationsdaten>

Rückgabe: -

```
{
  "model": "EWIO2-MW-BM",
  "serial": "00000018"
}
```

Bsp. für JSON-Struktur der Geräte-Konfiguration `device`

```
{
  "api-version": 1,
  "kernel-version": "4.9.88-ewio+g5ed2ee422",
  "package-version": "0.6",
  "build-number": "5f85065d"
}
```

Bsp. für JSON-Struktur der Geräte-Konfiguration `version`

```
{
  "datetime_utc": "2020-04-23 08:35:57",
  "datetime_local": "2020-04-23 10:35:57",
  "time_zone": "Europe\\Berlin",
  "time_server_1": "0.de.pool.ntp.org",
  "time_server_2": "1.de.pool.ntp.org",
  "utc_timestamp": 0
}
```

Bsp. für JSON-Struktur der Geräte-Konfiguration `datetime`

Besonderheiten beim Speichern der Geräte-Konfiguration `datetime`:

- Werden `time_server_...`-Felder angegeben, dann werden die Angaben zur aktuellen Zeit ignoriert.
- Zur Einstellung einer aktuellen Zeit muss das Feld `datetime_utc` verwendet werden (das Feld `datetime_local` wird ignoriert).
- Nach dem Speichern der Geräte-Konfiguration `datetime` sollte die API-Session durch einen Reboot des EWIO₂ beendet werden (siehe Abschnitt 3.1.4), um alle Systemkomponenten auf die neue Konfiguration einzustellen.

```
{
  "model": "EWIO2-MW-BM",
  "devicename": "EWIO2-a58649",
  "mac_address": "70:b3:d5:a5:86:49",
  "serialnumber": "00000018",
  "package_version": "0.6",
  "build_number": "5f85065d",
  "v_kernel": "4.9.88-ewio+g5ed2ee422",
  "flash_size": 3625360,
  "flash_free": 3536248
}
```

Bsp. für JSON-Struktur der Geräte-Konfiguration `devicebase`

```
{
    "sdcard_available":true,
    "sdcard_format":"vfat",
    "flash_size":3625360,
    "flash_free":3536240,
    "sd_size":3922944,
    "sd_free":3922912
}
```

Bsp. für JSON-Struktur der Geräte-Konfiguration `memory`

```
; IO Driver Configuration, automatically generated

FileWriteCount = 2

[device]
VerHardware IO = 1
VerHardware REL = 3

[relay outputs]
RelHandEnable = 15
RelHandActive = 0
RelHandValue = 0
RelDefault = 0

[transistor outputs]
DoHandEnable = 15
DoHandActive = 0
DoHandValue = 0
DoDefault = 0

[digital inputs]
DiDebounce 0 = 8
DiDebounce 1 = 8
DiDebounce 2 = 8
DiDebounce 3 = 8
DiDebounce 4 = 8
DiDebounce 5 = 8
DiDebounce 6 = 8
DiDebounce 7 = 8

[analog outputs]
AoHandEnable = 7
AoHandActive = 0
AoHandValue 0 = 0
AoHandValue 1 = 0
AoHandValue 2 = 0
AoHandValue 3 = 0
AoDefault 0 = 0
AoDefault 1 = 0
AoDefault 2 = 0
AoDefault 3 = 0

[analog inputs]
AiRange 0 = 1
AiRange 1 = 1
AiRange 2 = 1
AiRange 3 = 0
AiSensor 0 = 0
AiSensor 1 = 0
AiSensor 2 = 0
AiSensor 3 = 0

[sensor characteristics]
SensorTable 0
```

Bsp. für Text-Struktur der Geräte-Konfiguration `io_driver`

3.4 Funktionen zu Applikationen

3.4.1 Applikationsliste laden

Der Client fordert die Liste der auf dem EWIO₂ vorliegende Applikationen an. Der Parameter "module" wird nicht verwendet und kann leer sein oder auch weggelassen werden. Ergebnis ist die Liste der vorliegenden Applikationen im JSON-Format.

Methode: GET

URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:

username=Administrator&password=<tan-md5-hash>&type=applications&module=

Rückgabe: <JSON-Struktur>

```
{
  "apps":
  [
    "example1",
    "example2"
  ]
}
```

Bsp. für JSON-Struktur einer Applikationsliste

3.4.2 Applikation laden

Der Client fordert eine auf dem EWIO₂ vorliegende Applikation an. Die Applikation wird als Parameter "modul" ohne Extension angegeben.

(Bsp.: SOM_convert → modul= SOM_convert)

Die Applikation hat eine spezielle Struktur und liegt im JSON-Format vor.

Methode: GET

URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:

username=Administrator&password=<tan-md5-hash>&type=application&module=<Applikation>

Rückgabe: <JSON-Struktur>

3.5 M-Bus- Funktionen

3.5.1 Zähler-Liste laden

Vom Client wird eine Liste der konfigurierten M-Bus-Zähler angefordert.

Methode: GET

URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:

username=Administrator&password=<tan-md5-hash>&type=counter_list&
module=mbus_add_auto

Rückgabe: <JSON-Struktur>

```
{
  "counter":
  [
    {"ID":3,"BusAdr":"19235000-SYS-10-14","Name":"19235000-SYS-10-14"},
    {"ID":13,"BusAdr":"01040904-GAV-90-2","Name":"01040904-GAV-90-2"},
    {"ID":15,"BusAdr":"19235200-SYS-10-14","Name":"19235200-SYS-10-14"}
  ]
}
```

Bsp. für JSON-Struktur einer Zähler-Liste

3.5.2 Zählerkonfiguration löschen

Der angegeben Zähler soll aus der Konfiguration für M-Bus-Zähler auf dem EWIO₂ gelöscht werden.

Die zu löschenden Zähler müssen in einer speziellen JSON-Struktur angegeben werden:

{"counter": [<Counter Bus addr_1>, < Counter Bus addr_2>, ...>]}

Methode: POST

URL: <server-ip>/cgi-bin/setParamToServer.cgi

POST-Parameter:

Host:<server-ip>

Content-Type:application/x-www-form-urlencoded

Content-Length:<length>

Parameter:

username=Administrator&password=<tan-md5-hash>&type=counter_remove&
module=mbus_remove&data=<JSON-Struktur>

Rückgabe: -

```
data={
  "counter": [
    "19235200-SYS-10-14"
  ]
}
```

Bsp. für JSON-Struktur einer Liste für zu löschende Zähler

3.5.3 Zähler-Konfiguration speichern

Die Konfiguration eines M-Bus-Zählers wird gespeichert. Diese muss in einem speziellen JSON-Format vorliegen.

Methode: POST
URL: <server-ip>/cgi-bin/setParamToServer.cgi

POST-Parameter:

Host:<server-ip>

Content-Type:application/x-www-form-urlencoded

Content-Length:<length>

Parameter:

username=Administrator&password=<tan-md5-hash>&type=counter&
module=<Zähler-Busadr>&data=<JSON-Struktur>

Rückgabe: -

3.5.4 Zähler-Konfiguration laden

Vom Client wird die Konfiguration des angegebenen M-Bus-Zählers angefordert.

Methode: GET
URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:

username=Administrator&password=<tan-md5-hash>&type=counter&
module=<Zähler-Busadr>

Rückgabe: <JSON-Struktur>

```
data={
  "ID":1,
  "BusAdr":"19235200-SYS-10-14",
  "PrimAdr":10,
  "VerwPrim":1,
  "Konfiguration":"",
  "LProfDP":"",
  "BR":9600,
  "Name":"19235200-SYS-10-14",
  "Sort":1,
  "ZNummer":"123456",
  "Mandant":"",
  "AbnNr":"",
  "Gewerk":0,
  "Hersteller":"",
  "Konto":"",
  "Ort":"",
  "Kommentar":"",
  "btype":"MBus",
  "ztype":"",
  "meteringcode":{
    "Land": "DE",
    "Betreiber": 123456,
    "Plz": 12345,
    "MePuId": "EXAMPLE0123456789012"
  }
}
```

Bsp. für JSON-Struktur einer Zähler-Konfiguration

3.5.5 Datenpunktliste laden

Die Liste der Datenpunkte eines M-Bus-Zählers wird angefordert. Diese wird in einem speziellen JSON-Format angegeben.

Methode: GET

URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:

username=Administrator&password=<tan-md5-hash>&type=channels&
module=<Zähler-Busadr>

Rückgabe: <JSON-Struktur>

```
{
  "channel": [
    {
      "ID":1,
      "Zaehler_ID":1,
      "Freeze_ID":null,
      "Obis_ID": "",
      "Faktor_U":1,
      "Faktor_I":1,
      "EAblesung":"2020-06-05 11:20:00",
      "Intervall":5,
      "ZwTyp":"Volts",
      "MEinheit":"1.000000 V",
      "MessTyp":"normal",
      "Primaer":0,
      "App": "",
      "AppArgs": "",
      "Faktor":1,
      "Konfiguration": "",
      "Tele":1,
      "Rec":1
    },
    {
      "ID":2,
      "Zaehler_ID":1,
      "Freeze_ID":null,
      "Obis_ID": "",
      "Faktor_U":1,
      "Faktor_I":1,
      "EAblesung":"2020-06-05 11:20:00",
      "Intervall":5,
      "ZwTyp":"Energy",
      "MEinheit":"10 Wh",
      "MessTyp":"normal",
      "Primaer":0,
      "App": "",
      "AppArgs": "",
      "Faktor":1,
      "Konfiguration": "",
      "Tele":1,
      "Rec":2
    }
  ]
}
```

Bsp. für JSON-Format der Datenpunktliste

3.5.6 Datenpunkt-Konfiguration speichern

Die Konfiguration eines Datenpunktes von einem M-Bus-Zähler wird gespeichert. Diese muss in einem speziellen JSON-Format vorliegen.

Methode: POST
URL: <server-ip>/cgi-bin/setParamToServer.cgi

POST-Parameter:

Host: <server-ip>

Content-Type: application/x-www-form-urlencoded

Content-Length: <length>

Parameter:

username=Administrator&password=<tan-md5-hash>&type=channel&
module=<Zähler-Busadr>&data=<JSON-Struktur>

Rückgabe: -

3.5.7 Datenpunkt-Konfiguration laden

Die Konfiguration eines Datenpunktes von einem M-Bus-Zähler wird angefordert. Diese wird in einem speziellen JSON-Format angegeben.

Methode: GET
URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:

username=Administrator&password=<tan-md5-hash>&type=channel&
module=<Datenpunkt-ID>

Rückgabe: <JSON-Struktur>

```
{
  "channel": [
    {
      "ID":1,
      "Zaehler_ID":1,
      "Obis_ID":"",
      "Faktor_U":1,
      "Faktor_I":1,
      "EAblesung":"2020-06-05 11:20:00",
      "Intervall":5,
      "ZwTyp":"Volts",
      "MEinheit":"1.000000 V",
      "MessTyp":"normal",
      "Primaer":0,
      "App":"",
      "AppArgs":"",
      "Faktor":1,
      "Tele":1,
      "Rec":1
    }
  ]
}
```

Bsp. für JSON-Format der Datenpunkt-Konfiguration

3.5.8 M-Bus-Parameter für Dokumentation laden

Die Konfiguration eines Zählers mit sämtlichen dazugehörigen Datenpunkten wird als Textdatei ausgegeben.

Methode: GET
URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:
username=Administrator&password=<tan-md5-hash>&type=output&
module=mbus_<Zähler-Busadr>

Rückgabe: <Text>

```
Counter ID: 1
Query Number: 1
Counter Name: 00243216-ABB-32-2
MBUS-ID: 00243216-ABB-32-2
Baudrate: 2400
Counter Number:
Client:
Customer:
Craft: 0
Manufacturer:
Account:
Location:
Counter Type:
Bus Type: MBus
Meteringcode: DE12345612345EXAMPLE0123456789012
```

```
Data Points
  Channel ID: 1
  Telegram Number: 1
  Record Number: 1
  Freeze ID:
  OBIS ID:
  Readout From: 2020-04-16 11:15:00
  Interval: 5
  Description: Cumulation Counter
  Factor Voltage: 1
  Factor Current: 1
  Factor: 1
  Unit: 1.000000 V
  Prim. Counter: 0
  Application:
  Application Arguments:
```

Bsp. für Text-Format der Zähler-Konfiguration

3.5.9 M-Bus-Suche

Es wird eine Suche nach M-Bus-Zählern durchgeführt. Während der Suche werden zeilenweise Fortschrittmeldungen "PROGRESS_..." ausgegeben, das Ende der Suche wird durch eine Abschlusszeile "FOUND_<Anzahl Treffer>" angezeigt. Ein Abbruch der Suche (z.B. paralleler Start einer weiteren Suche) wird durch eine Abschlusszeile "ABORTED" angezeigt. Wenn die Suche wegen eines Fehlers (z.B. ungültige Suchparameter) nicht durchgeführt werden kann, wird nur eine Zeile "ERROR" zurückgegeben.

Methode: GET
URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:
 username=Administrator&password=<tan-md5-hash>&type=search&
 module=mbus_<Baudrate>[_Suchparameter]

Rückgabe: <Text>

Die Angabe der Suchparameter ist optional. Werden keine Suchparameter angegeben, wird eine Suche über alle Sekundäradressen durchgeführt. Der Suchparameter hat das Format <Adresstyp>[_Adressbereich], wobei der Adresstyp "sa" (für Suche über Sekundäradressen) oder "pa" (für Suche über Primäradressen) sein kann. Für Adresstyp "sa" kann optional als Adressbereich eine Sekundäradresse mit Wildcards angegeben werden. Für Adresstyp "pa" kann optional als Adressbereich eine Primäradresse (als einzelne Dezimalzahl) oder ein Primäradressbereich (als zwei durch Bindestrich getrennte Dezimalzahlen (<Anfang-Ende>)) angegeben werden.

Um an einer anderen Schnittstelle als der M-Bus-Schnittstelle zu suchen, kann dem Adressbereich ein Prefix "rs485:" (Suche an der RS-485-Schnittstelle) oder "<IP-Adresse>:<Port>:" (Suche über einen TCP-Gateway) vorangestellt werden.

```
PROGRESS_0FFFFFFFFFFFFFFF
PROGRESS_1FFFFFFFFFFFFFFF
PROGRESS_2FFFFFFFFFFFFFFF
PROGRESS_3FFFFFFFFFFFFFFF
PROGRESS_4FFFFFFFFFFFFFFF
PROGRESS_5FFFFFFFFFFFFFFF
PROGRESS_6FFFFFFFFFFFFFFF
PROGRESS_7FFFFFFFFFFFFFFF
PROGRESS_8FFFFFFFFFFFFFFF
PROGRESS_9FFFFFFFFFFFFFFF
FOUND_1
```

Bsp. für Suche über alle Sekundäradressen (Parameter module=mbus_2400)

```
PROGRESS_10
PROGRESS_11
PROGRESS_12
PROGRESS_13
PROGRESS_14
PROGRESS_15
FOUND_3
```

Bsp. für Suche über Primäradressen 10 bis 15 (Parameter module=mbus_2400_pa_10-15)

3.5.10 M-Bus-Suchergebnisse – Zählerliste laden

Vom Client wird eine Liste der in der letzten Suche gefundenen M-Bus-Zähler angefordert.

Methode: GET

URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:

username=Administrator&password=<tan-md5-hash>&type=counter_list&
module=mbus_search[_Baudrate]

Rückgabe: <JSON-Struktur>

Die Angabe der Baudrate ist optional. Wird eine Baudrate angegeben, werden nur bei der angegebenen Baudrate gefundene Zähler ausgegeben. Wird keine Baudrate angegeben, werden die letzten Suchergebnisse bei allen Baudraten ausgegeben.

```
{
  "counter":
  [
    { "BusAdr": "19235000-SYS-10-14", "PrimAdr": "10", "BR": 2400, "Name": "19235000-SYS-10-14" },
    { "BusAdr": "01040904-GAV-90-2", "PrimAdr": "11", "BR": 2400, "Name": "01040904-GAV-90-2" },
    { "BusAdr": "19235200-SYS-10-14", "PrimAdr": "12", "BR": 2400, "Name": "19235200-SYS-10-14" }
  ]
}
```

Bsp. für JSON-Struktur einer Zähler-Liste nach einer Suche bei 2400 Baud

3.5.11 M-Bus-Suchergebnisse – Datenpunktliste laden

Die Liste der Datenpunkte eines gefundenen M-Bus-Zählers wird angefordert. Diese wird in einem speziellen JSON-Format angegeben.

Methode: GET

URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:

username=Administrator&password=<tan-md5-hash>&type=channels&
module=mbus_search_<Zähler-Busadr>

Rückgabe: <JSON-Struktur>

```
{
  "channel": [
    {
      "Faktor_U":1,
      "Faktor_I":1,
      "EAblesung":"2020-06-05 11:20:00",
      "Intervall":0,
      "ZwTyp":"Volts",
      "MEinheit":"1.000000 V",
      "Primaer":0,
      "Faktor":1,
      "Tele":1,
      "Rec":1
    },
    {
      "Faktor_U":1,
      "Faktor_I":1,
      "EAblesung":"2020-06-05 11:20:00",
      "Intervall":0,
      "ZwTyp":"Energy",
      "MEinheit":"10 Wh",
      "Primaer":0,
      "Faktor":1,
      "Tele":1,
      "Rec":2
    }
  ]
}
```

Bsp. für JSON-Format der Datenpunktliste eines gefundenen Zählers

3.6 Modbus – Funktionen

3.6.1 Zählertyp - Liste laden

Es werden die Templates aller konfigurierbaren Zählertypen geladen. Die Zählertypen sind in einer speziellen JSON-Struktur enthalten.

Methode: GET

URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:

username=Administrator&password=<tan-md5-hash>&type=deviceType_list&
module=modbus_config

Rückgabe: <JSON-Struktur>

```
{
  "deviceTypes":
  [
    "modbus_schneider_Compact_NSX.json",
    "modbus_kbr_multimes 4F144-1-LED.json",
    "modbus_janitza_umg96s.json",
    "modbus_mc_mr-do4.json",
    "modbus_mc_mr-ai8_temperature.json",
    "modbus_mc_mr-ci4_current-0-20.json",
    "modbus_mc_mr-ai8_resistance.json",
    "modbus_schneider_PM3255.json",
    "modbus_schneider_iEM3250.json",
    "modbus_mc_mr-di10.json",
    "modbus_janitza_umg604.json",
    "modbus_janitza_umg503.json",
    "modbus_econ_sens+.json",
    "modbus_mc_mr-tp.json",
    "modbus_mc_mr-ci4_current-4-20.json",
    "modbus_mc_mr-ai8_voltage.json",
    "modbus_mc_mr-ci4_voltage.json",
    "modbus_mc_mr-si4.json",
    "modbus_mc_mr-di4.json",
    "modbus_schneider_iEM3255.json",
    "modbus_mc_mr-ao4.json"
  ]
}
```

Bsp. für JSON-Format der Zählertyp-Liste

3.6.2 Erweiterte Zählertyp - Liste laden

Es werden die Templates aller konfigurierbaren Zählertypen mit Template- und Zählertypnamen geladen. Die Zählertypen sind in einer speziellen JSON-Struktur enthalten.

Methode: GET
URL: <server-ip>/cgi-bin/getParamFromServer.cgi
Parameter:
 username=Administrator&password=<tan-md5-hash>&type=deviceType_object&
 module=modbus_config
Rückgabe: <JSON-Struktur>

```
{
  "deviceTypes":
  [
    {"template":"modbus_schneider_Compact_NSX.json","type":"schneider_NX100"},
    {"template":"modbus_kbr_multimess_4F144-1-LED.json","type":"kbr_multimess-4F144-1-LED"},
    {"template":"modbus_janitza_umg96s.json","type":"janitza_umg96s"},
    {"template":"modbus_mc_mr-do4.json","type":"mc_mr-do4"},
    {"template":"modbus_mc_mr-ai8_temperature.json","type":"mc_mr-ai8_temperature (Configuration 0x80 - 0x89)"},
    {"template":"modbus_mc_mr-ci4_current-0-20.json","type":"mc_mr-ci4 (Configuration 0..20mA)"},
    {"template":"modbus_mc_mr-ai8_resistance.json","type":"mc_mr-ai8_resistance (Configuration 0x40)"},
    {"template":"modbus_schneider_PM3255.json","type":"schneider_PM3255"},
    {"template":"modbus_schneider_iEM3250.json","type":"schneider_iEM3250"},
    {"template":"modbus_mc_mr-di10.json","type":"mc_mr-di10"},
    {"template":"modbus_janitza_umg604.json","type":"janitza_umg604 (without discrete harmonic)"},
    {"template":"modbus_janitza_umg503.json","type":"janitza_umg503"},
    {"template":"modbus_econ_sens+.json","type":"econ_sens+"},
    {"template":"modbus_mc_mr-tp.json","type":"mc_mr-tp"},
    {"template":"modbus_mc_mr-ci4_current-4-20.json","type":"mc_mr-ci4 (Configuration 4..20mA)"},
    {"template":"modbus_mc_mr-ai8_voltage.json","type":"mc_mr-ai8 (Configuration 0x00, default)"},
    {"template":"modbus_mc_mr-ci4_voltage.json","type":"mc_mr-ci4 (Configuration 0..10V)"},
    {"template":"modbus_mc_mr-si4.json","type":"mc_mr-si4"},
    {"template":"modbus_mc_mr-di4.json","type":"mc_mr-di4"},
    {"template":"modbus_schneider_iEM3255.json","type":"schneider_iEM3255"},
    {"template":"modbus_mc_mr-ao4.json","type":"mc_mr-ao4"}
  ]
}
```

Bsp. für JSON-Format der erweiterten Zählertyp-Liste

3.6.3 Zähler-Liste laden

Vom Client wird eine Liste der konfigurierten Modbus-Zähler angefordert.

Methode: GET

URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:

username=Administrator&password=<tan-md5-hash>&type=counter_list&
module= modbus_output

Rückgabe: <JSON-Struktur>

```
{
  "counter":
  [
    { "ID":14, "BusAdr":11, "Name":"mr do4" }
  ]
}
```

Bsp. für JSON-Struktur einer Zähler-Liste

Hinweis: Das JSON-Struktur-Feld „BusAdr“ kann sowohl numerisch (im Fall einer einfachen Modbus/RTU-Adresse) als auch als String (im Fall einer Modbus/TCP-Adresse) dargestellt sein. Anwendungen sollten mit beiden Darstellungsformen umgehen können.

3.6.4 Zählerkonfiguration löschen

Der angegeben Zähler soll aus der Konfiguration für Modbus-Zähler auf dem EWIO₂ gelöscht werden.

Die zu löschenden Zähler müssen in einer speziellen JSON-Struktur angegeben werden:

{"counter": [<Counter Bus addr_1>, < Counter Bus addr_2>, ...>]}

Methode: POST

URL: <server-ip>/cgi-bin/setParamToServer.cgi

POST-Parameter:

Host:<server-ip>

Content-Type:application/x-www-form-urlencoded

Content-Length:<length>

Parameter:

username=Administrator&password=<tan-md5-hash>&type=counter_remove&
module=modbus_remove&data=<JSON-Struktur>

Rückgabe: -

```
data={
  "counter": [
    "14"
  ]
}
```

Bsp. für JSON-Struktur einer Liste für zu löschende Zähler

3.6.5 Zähler-Konfiguration speichern

Die Konfiguration eines Modbus-Zählers wird gespeichert. Diese muss in einem speziellen JSON-Format vorliegen.

Methode: POST
URL: <server-ip>/cgi-bin/setParamToServer.cgi
POST-Parameter:
Host: <server-ip>
Content-Type: application/x-www-form-urlencoded
Content-Length: <length>
Parameter:
username=Administrator&password= <tan-md5-hash>&type=modbus_counter&
module= <Zähler-Busadr> &data= <JSON-Struktur>
Rückgabe: -

3.6.6 Zähler-Konfiguration laden

Vom Client wird die Konfiguration des angegebenen Modbus-Zähler angefordert.

Methode: GET
URL: <server-ip>/cgi-bin/getParamFromServer.cgi
Parameter:
username=Administrator&password= <tan-md5-hash>&type=modbus_counter&
module= <Zähler-Busadr>
Rückgabe: <JSON-Struktur>

```
{
  "ID":2,
  "BusAdr":6,
  "BR":0,
  "Name":"mr-ai8",
  "Sort":2,
  "ZNummer":"",
  "Mandant":"",
  "AbnNr":"",
  "Gewerk":0,
  "Hersteller":"",
  "Konto":"",
  "Ort":"",
  "Kommentar":"",
  "btype":"Modbus",
  "ztype":"mc_mr-ai8",
  "meteringcode": {
    "Land": "DE",
    "Plz": 12345,
    "Betreiber": 123456,
    "MePuId": "EXAMPLE0123456789012"
  }
}
```

Bsp. für JSON-Struktur einer Zähler-Konfiguration

Hinweis: Das JSON-Struktur-Feld „BusAdr“ kann sowohl numerisch (im Fall einer einfachen Modbus/RTU-Adresse) als auch als String (im Fall einer Modbus/TCP-Adresse) dargestellt sein. Anwendungen sollten mit beiden Darstellungsformen umgehen können.

3.6.6 Datenpunktliste laden

Die Liste der Datenpunkte eines Modbus-Zählers wird angefordert. Diese wird in einem speziellen JSON-Format angegeben.

Methode: GET
URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:
username=Administrator&password=<tan-md5-hash>&type=modbus_channels&module=<Zähler-Busadr>

Rückgabe: <JSON-Struktur>

```
{
  "channel": [
    {
      "ID":11,
      "Zaehler_ID":2,
      "Freeze_ID":null,
      "Rec":0,
      "Format":"INT16 HL",
      "Obis_ID":"",
      "Faktor_U":1,
      "Faktor_I":1,
      "EAblesung":"2020-06-05 11:20:00",
      "Intervall":1,
      "ZwTyp":"Volts",
      "MEinheit":"1.000000 V",
      "App":"",
      "AppArgs":"",
      "Faktor":1,
      "Konfiguration":""
    },
    {
      "ID":12,
      "Zaehler_ID":2,
      "Freeze_ID":null,
      "Rec":1,
      "Format":"INT16 HL",
      "Obis_ID":"",
      "Faktor_U":1,
      "Faktor_I":1,
      "EAblesung":"2020-06-05 11:20:00",
      "Intervall":1,
      "ZwTyp":"Input Resistance 1-C2",
      "MEinheit":"",
      "App":"",
      "AppArgs":"",
      "Faktor":1,
      "Konfiguration":""
    }
  ]
}
```

Bsp. für JSON-Format der Datenpunktliste

3.6.7 Datenpunkt-Konfiguration speichern

Die Konfiguration eines Datenpunktes von einem Modbus-Zähler wird gespeichert. Diese muss in einem speziellen JSON-Format vorliegen.

Methode: POST
URL: <server-ip>/cgi-bin/setParamToServer.cgi

POST-Parameter:
Host:<server-ip>
Content-Type:application/x-www-form-urlencoded
Content-Length:<length>

Parameter:
username=Administrator&password=<tan-md5-hash>&type=modbus_channel&
module=<Zähler-Busadr>&data=<JSON-Struktur>

Rückgabe: -

3.6.8 Datenpunkt-Konfiguration laden

Die Konfiguration eines Datenpunktes von einem Modbus-Zähler wird angefordert. Diese muss in einem speziellen JSON-Format vorliegen.

Methode: GET
URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:
username=Administrator&password=<tan-md5-hash>&type=modbus_channel&
module=<Datenpunkt-ID>

Rückgabe: <JSON-Struktur>

```
{
  "channel": [
    {
      "ID":12,
      "Zaehler_ID":2,
      "Obis_ID":"","
      "Rec":0,
      "Format":"INT16 HL",
      "Faktor_U":1,
      "Faktor_I":1,
      "EAblesung":"2020-06-05 14:15:00",
      "Intervall":5,
      "ZwTyp":"Input Resistance 1-C2",
      "MEinheit":"","
      "App":"","
      "AppArgs":"","
      "Faktor":1,
      "Konfiguration":""
    }
  ]
}
```

Bsp. für JSON-Format der Datenpunkt-Konfiguration

3.6.9 Modbus-Parameter für Dokumentation laden

Die Konfiguration eines Zählers mit sämtlichen dazugehörigen Datenpunkten wird als Textdatei ausgegeben.

Methode: GET
URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:
username=Administrator&password=<tan-md5-hash>&type=output&
module=modbus_<Zähler-Busadr>

Rückgabe: <Text>

```
Counter ID: 2
Query Number: 2
Counter Name: mr-ai8
Modbus Address: 6
Counter Number:
Client:
Customer:
Craft: 0
Manufacturer:
Account:
Location:
Counter Type: mc_mr-ao4
Comment:
Bus Type: Modbus
Meteringcode: DE12345612345EXAMPLE0123456789013
```

Data Points

```
Channel ID: 11
Freeze ID:
OBIS ID:
Record Number: 0
Readout From: 2020-04-16 11:15:00
Interval: 1
Description: Volts
Factor Voltage: 1
Factor Current: 1
Factor: 1
Unit: 1.000000 V
Application:
Application Arguments:
```

```
Channel ID: 12
Freeze ID:
OBIS ID:
Record Number: 1
Readout From: 2020-04-16 11:15:00
Interval: 1
Description: Input Resistance 1-C2
Factor Voltage: 1
Factor Current: 1
Factor: 1
Unit:
Application:
Application Arguments:
```

Bsp. für Text-Format der Zähler-Konfiguration

3.7 Systemzähler – Funktionen

3.7.1 Zähler-Liste laden

Vom Client wird eine Liste der konfigurierten System-Zähler angefordert.

Methode: GET
URL: <server-ip>/cgi-bin/getParamFromServer.cgi
Parameter:
username=Administrator&password=<tan-md5-hash>&type=counter_list&
module= system_output
Rückgabe: <JSON-Struktur>

```
{
  "counter":
  [
    { "ID": 20, "Name": "DI1" }
  ]
}
```

Bsp. für JSON-Struktur einer Zähler-Liste

3.7.2 Zählerkonfiguration löschen

Der angegeben Zähler soll aus der Konfiguration für System-Zähler auf dem EWIO₂ gelöscht werden.

Die zu löschenden Zähler müssen in einer speziellen JSON-Struktur angegeben werden:

{"counter": [<name_1>, <name_2>, ...>]}

Methode: POST
URL: <server-ip>/cgi-bin/setParamToServer.cgi
POST-Parameter:
Host:<server-ip>
Content-Type:application/x-www-form-urlencoded
Content-Length:<length>
Parameter:
username=Administrator&password=<tan-md5-hash>&type=counter_remove&
module=system_remove&data=<JSON-Struktur>
Rückgabe: -

```
data={
  "counter": [
    "DI1"
  ]
}
```

Bsp. für JSON-Struktur einer Liste für zu löschende Zähler

3.7.3 Zähler-Konfiguration speichern

Die Konfiguration eines System-Zählers wird gespeichert. Diese muss in einem speziellen JSON-Format vorliegen.

Methode: POST
URL: <server-ip>/cgi-bin/setParamToServer.cgi
POST-Parameter:
Host:<server-ip>
Content-Type:application/x-www-form-urlencoded
Content-Length:<length>
Parameter:
username=Administrator&password=<tan-md5-hash>&type=system_counter&
module=<Zähler-Name>&data=<JSON-Struktur>
Rückgabe: -

3.7.4 Zähler-Konfiguration laden

Vom Client wird die Konfiguration des angegebenen System-Zähler angefordert.

Methode: GET
URL: <server-ip>/cgi-bin/getParamFromServer.cgi
Parameter:
username=Administrator&password=<tan-md5-hash>&type=system_counter&
module=<Zähler-Name>
Rückgabe: <JSON-Struktur>

```
{
  "ID":20,
  "BR":0,
  "Name":"DI1",
  "Sort":3,
  "ZNummer":"",
  "Mandant":"",
  "AbnNr":"",
  "Gewerk":0,
  "Hersteller":"",
  "Konto":"",
  "Ort":"",
  "Kommentar":"",
  "btype":"System",
  "meteringcode": {
    "Land": "DE",
    "Plz": 12345,
    "Betreiber": 123456,
    "MePuId": "EXAMPLE0123456789012"
  }
}
```

Bsp. für JSON-Struktur einer Zähler-Konfiguration

3.7.5 Datenpunktliste laden

Die Liste der Datenpunkte eines System-Zählers wird angefordert. Diese wird in einem speziellen JSON-Format angegeben.

Methode: GET

URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:

username=Administrator&password=<tan-md5-hash>&type=system_channels&module=<Zähler-Name>

Rückgabe: <JSON-Struktur>

```
{
  "channel": [
    {
      "ID":21,
      "Zaehler_ID":20,
      "Freeze_ID":null,
      "Signal":"di_00_0",
      "Obis_ID":"",
      "Faktor_U":1,
      "Faktor_I":1,
      "EAblesung":"2021-06-05 11:20:00",
      "Intervall":1,
      "ZwTyp":"Datenpunkt 1",
      "MEinheit":"",
      "App":"",
      "AppArgs":"",
      "Faktor":1,
      "Konfiguration":""
    }
  ]
}
```

Bsp. für JSON-Format der Datenpunktliste

3.7.6 Datenpunkt-Konfiguration speichern

Die Konfiguration eines Datenpunktes von einem System-Zähler wird gespeichert. Diese muss in einem speziellen JSON-Format vorliegen.

Methode: POST
URL: <server-ip>/cgi-bin/setParamToServer.cgi

POST-Parameter:
Host: <server-ip>

Content-Type: application/x-www-form-urlencoded
Content-Length: <length>

Parameter:
username=Administrator&password=<tan-md5-hash>&type=system_channel&
module=<Zähler-Name>&data=<JSON-Struktur>

Rückgabe: -

3.7.7 Datenpunkt-Konfiguration laden

Die Konfiguration eines Datenpunktes von einem System-Zähler wird angefordert. Diese muss in einem speziellen JSON-Format vorliegen.

Methode: GET
URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:
username=Administrator&password=<tan-md5-hash>&type=system_channel&
module=<Datenpunkt-ID>

Rückgabe: <JSON-Struktur>

```
{
  "channel": [
    {
      "ID":21,
      "Zaehler_ID":20,
      "Signal":"di_00_0",
      "Obis_ID":"",
      "Faktor_U":1,
      "Faktor_I":1,
      "EAblesung":"2021-06-05 14:15:00",
      "Intervall":1,
      "ZwTyp":"Datenpunkt 1",
      "MEinheit":"",
      "App":"",
      "AppArgs":"",
      "Faktor":1,
      "Konfiguration":""
    }
  ]
}
```

Bsp. für JSON-Format der Datenpunkt-Konfiguration

3.7.8 Systemzähler-Parameter für Dokumentation laden

Die Konfiguration eines Zählers mit sämtlichen dazugehörigen Datenpunkten wird als Textdatei ausgegeben.

Methode: GET

URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:

username=Administrator&password=<tan-md5-hash>&type=output&
module=system_<Zähler-Name>

Rückgabe: <Text>

```
Counter ID: 20
Query Number: 1
Counter Name: DI1
Counter Number:
Client:
Customer:
Craft: 0
Manufacturer:
Account:
Location:
Comment:
Bus Type: System
Meteringcode: DE12345612345EXAMPLE0123456789013
```

```
Data Points
  Channel ID: 21
  Freeze ID:
  OBIS ID:
  Signal: di_00_0
  Readout From: 2021-04-16 11:15:00
  Interval: 1
  Description: Datenpunkt 1
  Factor Voltage: 1
  Factor Current: 1
  Factor: 1
  Unit:
  Application:
  Application Arguments:
```

Bsp. für Text-Format der Zähler-Konfiguration

3.8 Allgemeine Zählerfunktionen

3.8.1 Live-Check

Vom Client wird eine Überprüfung der Kommunikationsverbindung mit einem konfigurierten Zähler angefordert.

Methode: GET
URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:
username=Administrator&password=<tan-md5-hash>&type=testalive&
module=<Zähler-ID>

Rückgabe: <Text>

Die Zähler-ID ist der ID-Wert des Zählers, für den die Kommunikationsverbindung überprüft werden soll. Die Rückgabe ist *ACK*, falls die Kommunikationsverbindung zum Zähler fehlerfrei funktioniert, *NACK* falls ein Problem mit der Kommunikationsverbindung zum Zähler besteht (Zähler antwortet nicht) oder *ERROR* falls die Kommunikationsverbindung nicht überprüft werden kann (Zählerabfragedienst nicht verfügbar).

3.9 SQL- Funktionen, allgemein

3.9.1 Eine SQL-Anweisung senden

Vom Client wird eine SQL-Anweisung beliebigen Inhalts an den Server gesendet. Die Art der Antwort wird vom Parameter "type" festgelegt:

db_str = Antwort sofort als Textstring

db_json = Antwort sofort im JSON-Format

db_str_<nummer> = Antwort in nachfolgendem Funktionsaufruf als Textstring

db_json_<nummer> = Antwort in nachfolgendem Funktionsaufruf im JSON-Format

<nummer> = beliebige Zufallszahl, unter welcher das Resultat der Anweisung zur Verfügung steht

Der Parameter "module" wird nicht verwendet und kann leer sein oder auch weggelassen werden.

Methode: POST
URL: <server-ip>/cgi-bin/setParamToServer.cgi

POST-Parameter:

Host:<server-ip>

Content-Type:application/x-www-form-urlencoded

Content-Length:<length>

Parameter:

username=Administrator&password=<tan-md5-hash>&type=db_str_1234&
module=&data=<SQL-Statement>

Rückgabe: <Daten> (nur falls Antwort sofort)

3.9.2 Das Resultat einer SQL-Anweisung erhalten

Das Resultat einer mit Antwort in nachfolgendem Funktionsaufruf gesendeten SQL-Anweisung steht in einer temporären Datei zur Verfügung. Nach Abholung der Daten wird die Datei automatisch gelöscht.

Sind die Daten zum Abholungszeitpunkt noch nicht verfügbar, wird "WEBGATE ERROR – NO FILE" zurückgegeben. Der Aufruf ist dann nach einer gewissen Zeit zu wiederholen.

Methode: GET
URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:

username=Administrator&password=<tan-md5-hash>&type=db_str_1234&module=

Rückgabe: <Daten>

3.10 SQL- Funktionen, speziell

3.10.1 Messwerte aus Datenbank laden

Messwerte eines bestimmten Datenpunktes in einem angegebenen Zeitbereich werden angefordert. Ist keine Anzahl angegeben bzw. eine nicht gültige Anzahl angegeben, werden standardmäßig 24 Messwerte geliefert (falls vorhanden).

Als Modul wird der Datenpunkt – Unterstrich – Zeitbereich – Unterstrich – Anzahl angegeben. Der Datenpunkt wird durch seine Datenpunkt-ID angegeben.

Zeitbereich: -1 = letzte Daten
 0 = erste Daten
 1 = Daten ab (1 | <Datum Uhrzeit>)

Anzahl entspricht der Anzahl angeforderter Messwerte (standardmäßig 24). Zu beachten ist, dass eine große bis sehr große Anzahl an Messwerten u.U. lange Bereitstellungszeit benötigt bzw. gar nicht bereitgestellt werden kann!

Methode: GET
 URL: <server-ip>/cgi-bin/getParamFromServer.cgi

Parameter:
 username=Administrator&password=<tan-md5-hash>&type=db_data&
 module=<Datenpunkt-ID_Zeitbereich_Anzahl>

Rückgabe: <JSON-Struktur>

```
{
  "data":
  { "channelData":
    [
      { "Kanal_ID":2, "Zeit":"2020-05-08
      11:15:00", "Werte":0.000000, "Flags":"S;A;P;G;N;I;T", "Grund":"" },
      { "Kanal_ID":2, "Zeit":"2020-05-08
      11:20:00", "Werte":0.000000, "Flags":"S;A;P;G;N;I;T", "Grund":"" },
      { "Kanal_ID":2, "Zeit":"2020-05-08
      11:25:00", "Werte":0.000000, "Flags":"S;A;P;G;N;I;T", "Grund":"" },
      { "Kanal_ID":2, "Zeit":"2020-05-08
      11:30:00", "Werte":0.000000, "Flags":"S;A;P;G;N;I;T", "Grund":"" },

      ....

      { "Kanal_ID":2, "Zeit":"2020-05-08
      13:05:00", "Werte":0.000000, "Flags":"S;A;P;G;N;I;T", "Grund":"" },
      { "Kanal_ID":2, "Zeit":"2020-05-08
      13:10:00", "Werte":0.000000, "Flags":"S;A;P;G;N;I;T", "Grund":"" }
    ]
  }
}
```

Bsp. für JSON-Format der angeforderten Messwerte

3.10.2 Messwerte in Datenbank speichern

Ein oder mehrere Messwerte werden in der Datenbank auf dem EWIO₂ gespeichert. Diese müssen in einem speziellen JSON-Format vorliegen.

Methode: POST
URL: <server-ip>/cgi-bin/setParamToServer.cgi

POST-Parameter:

Host:<server-ip>

Content-Type:application/x-www-form-urlencoded

Content-Length:<length>

Parameter:

username=Administrator&password=<tan-md5-hash>&type=db_send&
module=mbus&data=<JSON-Struktur>

Rückgabe: -

```
{
  "channelData":
  [
    {
      "Kanal_ID":93,
      "Zeit":"2020-06-05 13:45:59",
      "Werte":1682,
      "Flags":"S;A;P;G;N;I;T",
      "Grund":""}
  ]
}
```

Bsp. für JSON-Format der zu speichernden Messwerte

4 Aufbau der Datenbank

Die Datenbankstruktur des EWIO₂ wird durch folgende SQL-Statements erzeugt (Korrespondenzen bezeichnen äquivalente Datenbankfelder des Vorgängergeräts EWIO-M):

```
CREATE TABLE `Counter` ( -- Basic properties of counter devices
    -- Unique ID of counter, corresponds to t_zaeehler.ID
    `znr` INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
    -- Secondary address of M-bus counter, unused for other devices, corresponds to t_zaeehler.BusAdr
    `address` TEXT,
    -- Primary address of M-bus counter or Modbus address of Modbus counter, corresponds to
    t_zaeehler.BusAdr
    `prim` INTEGER,
    -- Name of counter (assigned by user), corresponds to t_zaeehler.Name
    `name` TEXT,
    -- Bus type of counter (can be MBUS, MODBUS or SYSTEM), corresponds to t_zaeehler.BTyp_ID and
    t_btype.Kinds
    `bus` TEXT,
    -- Type of counter (selected by user, links to ID of row of MBusCounterTypes table for M-bus
    counter, name of Modbus-template for Modbus counter), corresponds to t_zaeehler.ZTyp_ID and
    t_ztype.Kinds
    `devicetype` TEXT,
    -- Baud rate of M-bus counter, unused for other devices, corresponds to t_zaeehler.BR, default I/O
    line for system counters
    `baud` INTEGER,
    -- Rank of counter for measurement value retrieval, measurement values from counter with lowest
    rank are retrieved first, corresponds to t_zaeehler.Sort
    `rank` INTEGER,
    -- Use secondary addressing instead of primary addressing for M-bus counter, unused for other
    devices
    `use_sec` INTEGER,
    -- Additional counter configuration data, number of load profile records to read out for EMH-DIZ
    counters, unused for other devices
    `config` TEXT,
    -- Index of first load profile data point for EMH-DIZ counters, unused for other devices
    `first_lp_dp` INTEGER,
    -- Time of last real-time clock synchronization for EMH-DIZ counters, unused for other devices
    `last_sync` TEXT
);

CREATE TABLE `Counter_extension` ( -- Extended informational properties of counter
    devices
    -- Unique ID of counter, links to related row of Counter table
    `znr` INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
    -- Craft property of counter (assigned by user)
    `craft` INTEGER,
    -- Counter number property of counter (assigned by user), corresponds to t_zaeehler.ZNummer
    `counternumber` TEXT,
    -- Manufacturer property of counter (assigned by user)
    `manufacturer` TEXT,
    -- Location property of counter (assigned by user)
    `location` TEXT,
    -- Account property of counter (assigned by user)
    `account` TEXT,
    -- Client property of counter (assigned by user), corresponds to t_zaeehler.Mandant
    `client` TEXT,
    -- Customer property of counter (assigned by user), corresponds to t_zaeehler.AbnNr
    `customer` TEXT,
    -- Metering-code country property of counter (assigned by user), corresponds to
    t_meteringcode.Land
```

```

`mc_country` TEXT,
-- Metering-code operator property of counter (assigned by user), corresponds to
  t_meteringcode.Betreiber
`mc_operator` TEXT,
-- Metering-code postcode property of counter (assigned by user), corresponds to
  t_meteringcode.Plz
`mc_postcode` TEXT,
-- Metering-code counter number property of counter (assigned by user), corresponds to
  t_meteringcode.MePuId
`mc_counternumber` TEXT,
-- Comment property of counter (assigned by user)
`comment` TEXT
);

CREATE TABLE `Datapoints` ( -- Basic properties of data points
-- Unique ID of data point, corresponds to t_kanal.ID
`id` INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
-- Unique ID of counter containing data point, links to related row of Counter table, corresponds
  to t_kanal.Zaehler_ID
`znr` INTEGER,
-- Index of data point within its counter for M-Bus and System counters, enumerates data points of
  a counter starting at 1, value is fixed to -1 for Modbus counter data points
`dpnr` INTEGER,
-- Index of corresponding freeze data point within its counter, or 0 if data point does not have a
  freeze data point
`freeze_id` INTEGER,
-- Modbus register address of Modbus counter data point, I/O line for system counter data point,
  index of M-Bus response telegram the datapoint is located in for M-Bus counter data points,
  corresponds to t_kanal.Rec for Modbus data points and to t_kanal.Tele for M-Bus data points
`register` INTEGER,
-- Name of data point, corresponds to t_kanal.ZwTyp
`name` TEXT,
-- Factor value of data point, applied before database insertion of measurement values,
  corresponds to t_kanal.Faktor_edit
`factor` REAL,
-- Unit of data point, corresponds to t_kanal.Einheit_edit
`unit` TEXT,
-- Measurement value query frequency of data point, corresponds to t_kanal.Intervall
`range` INTEGER,
-- Time of last data point readout or configuration change
`last` INTEGER,
-- Unique ID of data point with range minimization for BACnet server
`bacnet_id` INTEGER
);

CREATE TABLE `Datapoints_extension` ( -- Extended properties of data points
-- Unique ID of data point, links to related row of Datapoints table
`id` INTEGER PRIMARY KEY UNIQUE,
-- Time of first data point readout, corresponds to t_kanal.EAblesung
`readout` TEXT,
-- Type of measurement processing for data point (0: normal, 1: average, 2: minimum, 3: maximum),
  corresponds to t_kanal.MessTyp
`measuring` INTEGER,
-- Primarity property of data point, corresponds to t_kanal.Primaer
`prim` INTEGER,
-- Voltage factor of data point, applied before database insertion of measurements for data point
  related to a voltage, corresponds to t_kanal.Faktor_U
`u` REAL,
-- Current factor of data point, applied before database insertion of measurements for data point
  related to a current, corresponds to t_kanal.Faktor_I

```

```
`i` REAL,
-- Data type of Modbus data point, unused for data points not belonging to a Modbus counter
`datatype` TEXT,
-- Temperature sensor type of data point, links to ID of row of TempSensorTypes table, corresponds
  to t_temp.SensTyp
`sens_type` INTEGER,
-- Offset to be applied after temperature sensor data transformation of measurement value,
  corresponds to t_temp.Offset
`sens_offset` REAL,
-- OBIS-ID of data point (assigned by user), corresponds to t_kanal.Obis_ID
`obis_id` TEXT,
-- Application script to be executed during measurement value retrieval, corresponds to
  t_kanal.App
`app` TEXT,
-- Arguments to be supplied to application script executed during measurement value retrieval
`app_args` TEXT,
-- Additional datapoint configuration data, a value > 0 indicates that a new measurement should be
  saved only if it differs from the last saved measurement
`config` TEXT
);
CREATE TABLE `MbusCounterTypes` ( -- Types of M-bus counters
  -- Unique ID of M-bus counter type
  `id` INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
  -- Name of M-bus counter type
  `name` TEXT
);
CREATE TABLE `TempSensorTypes` ( -- Types of temperature sensors
  -- Unique ID of temperature sensor, corresponds to t_sensor.ID
  `id` INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
  -- Name of temperature sensor type (shown to user in type selection list), corresponds to
    t_sensor.Name
  `name` TEXT
);
CREATE TABLE `ModbusCounterModels` ( -- Types (models) of Modbus counters
  -- Unique ID of Modbus counter type
  `model_id` INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
  -- Default Modbus address of Modbus counter type
  `address` INTEGER,
  -- Name of Modbus counter type
  `modelname` TEXT UNIQUE,
  -- Template of Modbus counter data points
  `template` TEXT);
CREATE TABLE `ModbusCounterModelsDatapoints` ( -- Default data points of Modbus
  counter types devices
  -- Unique ID of default Modbus counter data point
  `db_id` INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
  -- Unique ID of Modbus counter type, links to related row of ModbusCounterModels table
  `model_id` INTEGER,
  -- Index of default datapoint within a Modbus counter type, enumerates data points of a counter
    type starting at 1
  `dpnr` INTEGER,
  -- Modbus register address of default data point
  `register` INTEGER,
  -- Name of default data point
  `name` TEXT,
  -- Unit of default data point
```

```
`unit`      TEXT,
-- Data type of default data point
`format`    TEXT,
-- Default factor for data point
`factor`    TEXT,
-- Default obis_id for data point
`obis_id`   TEXT
);
CREATE TABLE `Measurements` ( -- Measurement values
-- Unique ID of measurement value, corresponds to t_daten.ID
`id` INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
-- Unique ID of data point that provided the measurement value, links to related row of Datapoints
  table, corresponds to t_daten.Kanal_ID
`dpid`      INTEGER,
-- Sample time of measurement value, corresponds to t_daten.Zeit
`sampletime` TEXT DEFAULT CURRENT_TIMESTAMP,
-- Value (result of all transformations applied during the readout process) of measurement value,
  corresponds to t_daten.Werte
`value`     REAL DEFAULT 0,
-- Flags of measurement value, corresponds to t_flags.Flags
`flags`     TEXT,
-- Additional description of measurement value
`reason`    TEXT
);
```