Programmers Guide

KU PM BB 001800 B
KU PM BB 0104000 B

Document Version: 1.0
Associated Software Version: 1.2
Date: April 13, 2023

1 SNMP-OID Overview

OID-Prefix: `1.3.6.1.4.1.56710.1`
RO-Community: `public`
SNMP Version v1 and v2 supported. Currently settings are not changeable over the SNMP-Interface. Use Web-Interface to change.

The OID-Suffix in the following table must be appended to the OID-Prefix. The full OID of *power1* is for example: `1.3.6.1.4.1.56710.1.1.1.0`
The OID-Table is also avalilable as a *.mib*-File

OID-Suffix	Name	Type	Syntax	Unit	Explanation
values					
1.1.0	power1	OCTET STRING	-999.99[1]	dBm	Measured Value of Port 1
1.2.0	power2	OCTET STRING	-999.99[1]	dBm	Measured Value of Port 2
1.3.0	timestamp	OCTET STRING	Datetime[2]		Date and Time of last Measurement
1.4.0	alarmStatus1	INTEGER	alarm[3]		Alarm status for Port 1
1.5.0	alarmStatus2	INTEGER	alarm[3]		Alarm status for Port 2
1.6.0	alarmStatusGlobal	INTEGER	alarm[3]		Combined alarm status
properties					
system					
device					
2.0.0.1.0	model	OCTET STRING			Factory Set Model ID
2.0.0.2.0	firmware	OCTET STRING			Installed Firmware Version
2.0.0.3.0	serialNumber	OCTET STRING	00000000		Factory Set Serial Number
description					
2.0.1.1.0	deviceName	OCTET STRING	ASCII(15)		User settable
2.0.1.2.0	deviceLocation	OCTET STRING	ASCII		User settable
2.0.1.3.0	devicePointOfContact	OCTET STRING	ASCII		User settable
network					
2.0.2.1.0	ip	IPADDRESS			IP-Address of Device
2.0.2.2.0	netmask	IPADDRESS			Netmask of Device
2.0.2.3.0	gateway	IPADDRESS			Gateway of Device
2.0.2.4.0	mac	OCTET STRING			MAC-Address of Device
misc					
2.0.3.1.0	onlineSince	OCTET STRING	Datetime[2]		Startup Date and Time
2.0.3.2.0	recordStatus	INTEGER	0/1		Status of datalogger
2.0.3.3.0	recordRunningSince	OCTET STRING	Datetime[2] or 'Not Running'		Start date/time of current record
2.0.3.4.0	recordDataPoints	INTEGER			Datapoints in current record
2.0.3.5.0	acquisitionMode	INTEGER	mean/max/min		Handling of the measured Points
2.0.3.6.0	maxFileLength	INTEGER			Maximum length of record file
2.0.3.7.0	displayColor	OCTET STRING	HEX-Colorcode		Normal display color
2.0.3.8.0	displayColorWarningLevel	OCTET STRING	HEX-Colorcode		Display color at warning
2.0.3.9.0	displayColorAlarmLevel	OCTET STRING	HEX-Colorcode		Display color at alarm
alarm					
2.0.4.1.0	snmpTrapActive	INTEGER	0/1		Alarm over SNMP-TRAP
2.0.4.2.0	snmpManagerIP	IPADDRESS			IP-Address of SNMP Manager
2.0.4.3.0	snmpManagerPort	INTEGER			Port of SNMP Manager
2.0.4.4.0	httpAlarmActive	INTEGER	0/1		Alarm over HTTP-Request
2.0.4.5.0	httpAlarmUrl	OCTET STRING	URL/IP		URL for HTTP-Request Alarms
2.0.4.6.0	httpRequestType	OCTET STRING	'GET'/'POST'		Request Method

2.0.4.4.0	alarmRate	INTEGER	0: No Repetition	sec	Rate of Alarm Repetition

	sysclock				
2.0.5.1.0	localTime	OCTET STRING	Datetime[5]		System time in local timezone
2.0.5.2.0	universalTime	OCTET STRING	Datetime[5]		System time in universal timezone
2.0.5.3.0	rtcTime	OCTET STRING	Datetime[6]		Time of RTC-Chip
2.0.5.4.0	timeZone	OCTET STRING	Time Zone [7]		Configured Timezone
2.0.5.5.0	ntpActive	INTEGER	0/1		Status of NTP-Service
2.0.5.6.0	ntpPrimary	OCTET STRING	URL/IP		URL of primary NTP-Server
2.0.5.7.0	ntpSecondary	OCTET STRING	URL/IP		URL of fallback NTP-Server
	port1settings				
2.1.1.0	port1Name	OCTET STRING	ASCII(15)		User settable
2.1.2.0	port1Frequency	INTEGER		MHz	User selectable
2.1.3.0	port1Offset	OCTET STRING	-99.99[1]	dBm	User settable. Is added to measured value
2.1.4.0	warningLevelLow1	OCTET STRING	-999.99[1]	dBm	Port 1 Low Warning Level
2.1.5.0	alarmLevelLow1	OCTET STRING	-999.99[1]	dBm	Port 1 Low Alarm Level
2.1.6.0	warningLevelHigh1	OCTET STRING	-999.99[1]	dBm	Port 1 High Warning Level
2.1.7.0	alarmLevelHigh1	OCTET STRING	-999.99[1]	dBm	Port 1 High Alarm Level
2.1.8.0	port1CalDatetime	OCTET STRING	Datetime[4]		Port 1 Date and time of active calibration
2.1.9.0	port1CalLocation	OCTET STRING			Port 1 Location of active calibration
	port2settings				
2.2.1.0	port2Name	OCTET STRING	ASCII(15)		User settable
2.2.2.0	port2Frequency	INTEGER		MHz	User selectable
2.2.3.0	port2Offset	OCTET STRING	-99.99[1]	dBm	User settable. Is added to measured value
2.2.4.0	warningLevelLow2	OCTET STRING	-999.99[1]	dBm	Port 2 Low Warning Level
2.2.5.0	alarmLevelLow2	OCTET STRING	-999.99[1]	dBm	Port 2 Low Alarm Level
2.2.6.0	warningLevelHigh2	OCTET STRING	-999.99[1]	dBm	Port 2 High Warning Level
2.2.7.0	alarmLevelHigh2	OCTET STRING	-999.99[1]	dBm	Port 2 High Alarm Level
2.2.8.0	port2CalDatetime	OCTET STRING	Datetime[4]		Port 2 Date and time of active calibration
2.2.9.0	port2CalLocation	OCTET STRING			Port 2 Location of active calibration

[1] No leading zero
[2] ok(0), alarm(1), warning(2)
[3] DDD, dd mmm yyyy HH:MM:SS e.g Wed, 21 Apr 2021 12:34:56
[4] DDD, dd mmm yyyy HH:MM e.g Wed, 21 Apr 2021 12:34
[5] DDD yyyy-mm-dd HH:MM:SS ZZZ e.g Wed 2021-04-21 12:34:56 UTC
[6] DDD yyyy-mm-dd HH:MM:SS e.g Wed, 2021-04-21 12:34:56
[7] region/city(UTC ±XXXX)

2 JSON Dataset Overview

All Settings and the current measured values can be queried in *.json* format over the URL:
`<ip>/data/full.json`

There is also a smaller dataset available which contains only the measured values, the timestamp and the alarm-status:
`<ip>/data/values.json`

JSON-Key	Type	Format	Unit	Explanation
power1	NUMBER		dBm	Measured Value of Port 1
power2	NUMBER		dBm	Measured Value of Port 2
timestamp	STRING	DDD, dd.mm.yyyy HH:MM:SS		Date and Time of last Measurement
alarmStatus1	NUMBER	ok(0), alarm(1), warning(2)		Alarm status for Port 1
alarmStatus2	NUMBER	ok(0), alarm(1), warning(2)		Alarm status for Port 2
alarmStatusGlobal	NUMBER	ok(0), alarm(1), warning(2)		Combined Alarm Status

Table 2: JSON Dataset `<ip>/data/values.json`

JSON-Key	Type	Format	Unit	Explanation
model	STRING			Factory Set Model ID
firmware	STRING			Installed Firmware Version
serialNumber	STRING			Factory Set Serial Number
deviceName	STRING	ASCII, MAX 15 CHAR		User settable
deviceLocation	STRING	ASCII		User settable
devicePointOfContact	STRING	ASCII		User settable
ip	STRING			IP-Address of Device
netmask	STRING			Netmask of Device
gateway	STRING			Gateway of Device
mac	STRING			MAC-Address of Device
onlineSince	STRING	DDD, dd mmm yyyy HH:MM:SS		Startup Date and Time
recordStatus	STRING	ACTIVE/INACTIVE		Status of datalogger
recordRunningSince	STRING	DDD, dd mmm yyyy HH:MM:SS or 'Not Running'		Start date/time of current record
recordDataPoints	NUMBER			Datapoints in current record
acquisitionMode	STRING	mean/max/min		Handling of the measured Points
maxFileLength	NUMBER			Maximum length of record file
displayColor	STRING	HEX COLOR CODE		Normal display color
displayColorWarningLevel	STRING	HEX COLOR CODE		Display color at warning
displayColorAlarmLevel	STRING	HEX COLOR CODE		Display color at alarm
snmpTrapActive	BOOL			Alarm over SNMP-TRAP
snmpManagerIP	STRING			IP-Address of SNMP Manager
snmpManagerPort	NUMBER			Port of SNMP Manager
httpAlarmActive	BOOL			Alarm over HTTP-Request
httpAlarmUrl	STRING			URL for HTTP-Request Alarms
httpRequestType	STRING	'GET'/'POST'		Request Method
alarmRate	NUMBER	0: No Repetition	sec	Rate of Alarm Repetition
localTime	STRING	DDD yyyy-mm-dd HH:MM:SS ZZZ		System time in local timezone
universalTime	STRING	DDD yyyy-mm-dd HH:MM:SS ZZZ		System time in universal timezone
rtcTime	STRING	DDD yyyy-mm-dd HH:MM:SS		Time of RTC-Chip
timeZone	STRING	region/city(UTC ±XXXX)		Configured Timezone
ntpActive	BOOL	0/1		Status of NTP-Service
ntpPrimary	STRING	URL/IP		URL of primary NTP-Server
ntpSecondary	STRING	URL/IP		URL of fallback NTP-Server
port1Name	STRING	ASCII, MAX 15 CHAR		User settable
port1Frequency	NUMBER	[MHz]	MHz	User selectable

port1Offset	NUMBER	[dBm]	dBm	User settable. Is added to measured value
warningLevelLow1	NUMBER	[dBm]	dBm	Port 1 Low Warning Level
alarmLevelLow1	NUMBER	[dBm]	dBm	Port 1 Low Alarm Level
warningLevelHigh1	NUMBER	[dBm]	dBm	Port 1 High Warning Level
alarmLevelHigh1	NUMBER	[dBm]	dBm	Port 1 High Alarm Level
port1CalDatetime	STRING	DDD, dd mmm yyyy HH:MM		Port 1 Date and time of active calibration
port1CalLocation	STRING			Port 1 Location of active calibration
port2Name	STRING	ASCII(15)		User settable
port2Frequency	NUMBER		MHz	User selectable
port2Offset	NUMBER		dBm	User settable. Is added to measured value
warningLevelLow2	NUMBER		dBm	Port 2 Low Warning Level
alarmLevelLow2	NUMBER		dBm	Port 2 Low Alarm Level
warningLevelHigh2	NUMBER		dBm	Port 2 High Warning Level
alarmLevelHigh2	NUMBER		dBm	Port 2 High Alarm Level
port2CalDatetime	STRING	DDD, dd mmm yyyy HH:MM		Port 2 Date and time of active calibration
port2CalLocation	STRING			Port 2 Location of active calibration

Table 3: JSON Dataset <ip>/data/full.json

3 Alarm Function

In addition to the possibility to query values of the Powermeter, it is also capable of sending notifications when predefined alarm levels are exceeded. The *KU PB BB 001800 A* supports notifications via SNMP-Trap and HTTP GET/POST Request. Alarm conditions, notification methods and alarm repetition rates can be configured on the "Settings" tab under "Alarm". Notifications are sent individually for each channel when an alarm condition is reached or when it changes from warning to alarm or vice versa. If the alarm repetition is activated, all active alarms will be resent at the selected interval.

3.1 SNMP-TRAP

SNMP-Traps are sent to the configured SNMP-Manager using *SNMP Version 2*. The Community is *public*.
Trap OID for Notifications on Port 1 is: `.1.3.6.1.4.1.56710.1.0.1`
Trap OID for Notifications on Port 2 is: `.1.3.6.1.4.1.56710.1.0.2`

Variable Bildings:

Name:	sysUpTime.0
Value:	[TimeTicks]
Name:	snmpTrapOID
Value:	[OID] .1.3.6.1.4.1.56710.1.0.1 (alarmPort1Notification) OR .1.3.6.1.4.1.56710.1.0.2 (alarmPort2Notification)
Name:	.1.3.6.1.4.1.56710.1.1.4.0 (alarmStatus1) OR .1.3.6.1.4.1.56710.1.1.5.0 (alarmStatus2)
Value:	[Integer] ok(0), alarm(1), warning(2)
Name:	1.3.6.1.4.1.56710.1.1.1.0 (power1) OR .1.3.6.1.4.1.56710.1.0.2 (power2)
Value:	[OctetString] -999.99

3.2 HTTP-Request

HTTP-Requests can either be sent via GET or POST Method. Both methods will send the same variables:

`source`	IP-Address of the Powermeter
`channel`	Number (1/2) of the respective channel
`level`	Severity of the Alarm: 1=Alarm, 2=Warning
`value`	Measured value of the respective channel

An HTTP-GET URL would look something like this:
`http://10.10.10.1/alarmReceiver.html?source=10.10.10.10&channel=1&level=2&value=-42.42`

4 Code examples

In the following section working code examples in *Python* are being presented to provide a quick start into using the API of the *KU PB BB 001800 A*. They are kept at a minimum to ensure readability.

4.1 Requesting Values

Values can be retrieved in *.json* fromat the device according to Section 2 using requests. The following example queries the measured value of both channels from the *KU PB BB 001800 A* and prints them to the console.

```
import requests

ip = "10.10.10.10"

response = requests.get("http://"+ip+"/data/values.json")
data = response.json()

print(data.get("power1"))
print(data.get("power2"))
```

4.2 Alarm Handling

For receiving the HTTP-Request alarms a Webserver is required. A simple way to do so in *Python* is using *Flask* which comes with *Werkzeug* as a development server. To set it up, it is only necessary to install *flask*, for example using *pip* and run one of the following scripts. The *KU PB BB 001800 A* must be configured to send the alarms as HTTP-Request to the respective computer's IP-Address on Port 5000, for example: `http://10.10.10.11:5000`.
Please note that a development server should not be used in production environments due to stability issues.

The first example receives alarms which are sent as a GET-Request and prints the content to the console.

```
from flask import Flask, request
app = Flask(__name__)

@app.route("/")
def get_test():
  source = request.args.get("source")
  channel = request.args.get("channel")
  level = request.args.get("level")
  value = request.args.get("value")

  print("Alarm sent via GET Method")
  print("Device-IP: " + source)
  print("Channel: " + channel)
  print("Level: " + level)
  print("Value: " + value)
  return ""

#Start the development server if file is executed directly
if __name__ == "__main__":
    app.run(host = "0.0.0.0", debug = True, port = 5000)
```

The following example does the same thing as the first one, but it handles alarms that are sent as a POST-Request.

```
from flask import Flask, request
app = Flask(__name__)

@app.route("/", methods = ["POST"])
def post_test():
  source = request.form.get("source")
  channel = request.form.get("channel")
  level = request.form.get("level")
  value = request.form.get("value")

  print("Alarm sent via GET Method")
  print("Device-IP: " + source)
  print("Channel: " + channel)
  print("Level: " + level)
  print("Value: " + value)
  return ""

#Start the development server if file is executed directly
if __name__ == "__main__":
  app.run(host = "0.0.0.0", debug = True, port = 5000)
```

The following example is a simple alarm handler that receives alarms as a HTTP-GET-Request, processes the included information and requests additional information from the device that sent the alarm. The information is printed to the console and saved to a log-file in the following format:

```
------------------------------------------------
Alarm Received: Mon, 31 May 2021 10:53:05
Device: 192.168.0.129 (Name: KUPMBB001800A)
Channel: 1 (Name: Port 1)
Level: 2 (Warning)
Value: -83.80 dBm
Time of PowerMeter: Mon, 31 May 2021 08:53:30
```

```python
from flask import Flask, request
import requests
import time

app = Flask(__name__)

@app.route("/")
def get_test():
  #Processing the arguments from the alarm request
  source = request.args.get("source")
  channel = request.args.get("channel")
  level = request.args.get("level")
  value = request.args.get("value")

  #Generating the local timestamp
  timestamp = time.strftime("%a, %d %b %Y %H:%M:%S", time.localtime())

  #Retreiving additional properties from the PowerMeter
  response = requests.get("http://"+source+"/data/full.json")
  pmProps = response.json()

  #Writing alarm data to console
  print("Alarm Received: " + timestamp)
  print("Time of PowerMeter: " + pmProps.get("timestamp"))
  print("Device: " + source + " (Name: " + pmProps.get("deviceName") + ")")
  print("Channel: " + channel + " (Name: " + pmProps.get("port"+str(channel)+"Name") + ")")
  print("Level: " + level + (" (Warning)" if level is "2" else " (Alarm)"))
  print("Value: " + value + " dBm")

  #Writing alarm data to log-file
  with open("alarms.log", "a") as logfile:
    logfile.write("------------------------------------------------\n")
    logfile.write("Alarm Received:  " + timestamp + "\n")
    logfile.write("Time of PowerMeter: " + pmProps.get("timestamp") + "\n")
    logfile.write("Device: " + source + " (Name: " + pmProps.get("deviceName") + ")\n")
    logfile.write("Channel: " + channel + " (Name: "+pmProps.get("port"+str(channel)+"Name")+")\n")
    logfile.write("Level: " + level + (" (Warning)" if level is "2" else " (Alarm)") + "\n")
    logfile.write("Value: " + value + " dBm" + "\n")

  return ""

#Start the development server if file is executed directly
if __name__ == "__main__":
  app.run(host = "0.0.0.0", debug=True, port=5000)
```