PoolLab2 BLE API Reference

Water-i.d. GmbH

January 30, 2024

Contents

1 Introduction

This document serves as reference for the PoolLab2's Bluetooth®Low Energy software API and provides examples for application developers.

2 Bluetooth®Profile

2.1 General

The PoolLab2 implements a Bluetooth®Low Energy 4.2 peripheral (slave) device. The PoolLab2 is scannable by any Bluetooth®Low Energy central (master) device while the radio is enabled. The device implements a single GATT `Service`, comprised of 3 `Characteristics`, that are usable as I/O for a command-and-response interface. This interface can be used to control, configure and update the PoolLab2, as well as download saved measurement data.

Note that the radio is disabled by default after reset and wakeup, so user interaction with the PoolLab2 is required to turn on the radio before a connection is possible. The way to achieve this can potentially differ for different OEM versions of the device, but in general the Bluetooth®Low Energy radio can be enabled by double-pressing the lower-right button. The Bluetooth®Logo on the PoolLab2's LCD display will turn green to indicate that the radio is enabled and the PoolLab2 is scannable and connectable by nearby Bluetooth®Low Energy central devices.

2.2 GAP Profile

An application should scan for PoolLab2 devices using their host platform's software interface, preferably using the address-based scan-method if available. The PoolLab2 advertises it's name and address. The Bluetooth®MAC-address starts with the hexadecimals [**60:44:7A**] and it is advertising it's name as **PoolLab2**.

The `Securitymode` is **Mode 1**, so there is no pairing code exchanged between the PoolLab2 and a central device. The PoolLab2 does not support **Bonding**. Storing pairing keys for longer than the duration of the connection can lead to connection problems on subsequent connections, as the stored pairing codes (the **Bonding**) could potentially have become invalid by that time.

The PoolLab2 can only be connected to a single central device at a time, and advertising is stopped when a connection from a central device is established. However, central devices that have cached scan-results can still attempt to connect while another central device is already connected. In this situation, the second connection attempt fails. Bluetooth®Low Energy advertisement is automatically resumed when the central device disconnects.

2.3 GATT Profile

The PoolLab2 implements a single and static `GATT Service` called **PoolLabSvc**, identified by UUID **{593FAE78-D97C-438D-92E4-FC082B5EC218}**. The **PoolLabSvc** must be discovered on every connection attempt by the host platform as the first step of the connection.

The **PoolLabSvc** is comprised of three `Characteristics`:

Description	UUID
MOSI_CMD	79989C85-B98E-4A73-A3AA-BA95E55E5EED0
MISO_CMD	0304B80F-FF49-4D59-9B7A-6C53F716C959
MISO_SIG	4E1765D2-8517-4A6A-A8A1-39D8FCBBD40C

Table 1: **PoolLabSvc**'s Client Characteristics Descriptors and their UUIDs

The **MOSI_CMD** `Characteristic` is intended for central devices to write commands and data into. It is 508 bytes wide. The PoolLab2 does not update this characteristic's value, it is only written to by central devices.

The **MISO_CMD** `Characteristic` is intended for central devices to read back additional data. It is 508 bytes wide, and the value is changed by the PoolLab2's firmware. Writes from the central device into this characteristic are ignored.

Finally, **MISO_SIG** is a read-only `Characteristic`, that is updated by the device after a command is processed. It is 16 bytes wide. As part of the connection process, a central device shall enable notifications of this `Characteristic` by using the host platform's API, or writing the hexadecimal value of `[0x0100]` into the `Descriptor` of this `Characteristic`.

3 Initiating a Connection

3.1 Enabling Bluetooth®

The first step is enable Bluetooth®Low Energy on the PoolLab2 as well as on the central device. The PoolLab2 indicates the status of it's Bluetooth®radio with a Bluetooth®symbol in the top-left corner of the LCD display. When the radio is enabled, the symbol is highlighted in green.

3.2 Bluetooth®Low Energy Scan

The next step is to initiate a Bluetooth®Low Energy peripheral device discovery ("scan") on the central device, and to filter for PoolLab2 devices by filtering for the Bluetooth®name and the manufacturer-part of the MAC-address (see section "GAP Profile"). When a PoolLab2 is found, the central can start a Bluetooth®Low Energy connection.

3.2.1 GAP Connection

The first step of the connection, the GAP protocol, is usually handled automatically and transparently to the application, and no special steps or configurations are required by the central-device application.

> "The GAP layer of the Bluetooth low energy protocol stack is responsible for connection functionality. This layer handles the access modes and procedures of the device including device discovery, link establishment, link termination, initiation of security features, and device configuration."[1]

The PoolLab2 implements the most simple form of this protocol, meaning that there is no pairing code exchanged between the central device and the PoolLab2, and there can be only one central device connected to the PoolLab2 at a time. Using different security/access modes is possible on request.

3.3 GATT Service Discovery

Next, the central device shall scan the peripheral for the list of it's GATT services, and find the **PoolLabSvc** identified by it's unique-ID (see section "GATT Profile"). Once the service is discovered, a "detail discovery" must be performed on the **PoolLabSvc**.

3.4 Enabling Notifications

Finally, the central device shall update the `Descriptor` of the **MISO_SIG** `Characteristic` (see above) to enable notifications on the GATT layer. This step is required for each connection attempt to implement the Command-and-Response logic described in the next section of this document.

At this point, the logical connection is ready and commands can be issued by the central device to the PoolLab2. For more information on how to use this interface, see the next section.

3.5 Recommended first Commands

After the low-level connection is established, it is highly recommended to always send the `GET_BATTERY_VOLTAGE` command first and verify the device's battery voltage is above 3700mV. A device with a battery voltage below this could decide to switch off it's radio or even enter sleep-mode in the very near future, so it is recommended to disconnect from the PoolLab2 in this case.

Another highly recommended next step, after verifying the battery-voltage is above 3700mV, is then to send the `GET_QUICK_INFO` command, which will return almost all relevant information about the device with a single command, and most importantly, the firmware version of the PoolLab2.

As the firmware evolves, new functions and commands are added, which means some commands are unavailable with older firmware versions. A list of commands and their required minimum firmware versions are given in the next sections in this document.

4 Command and Response Interface

4.1 General

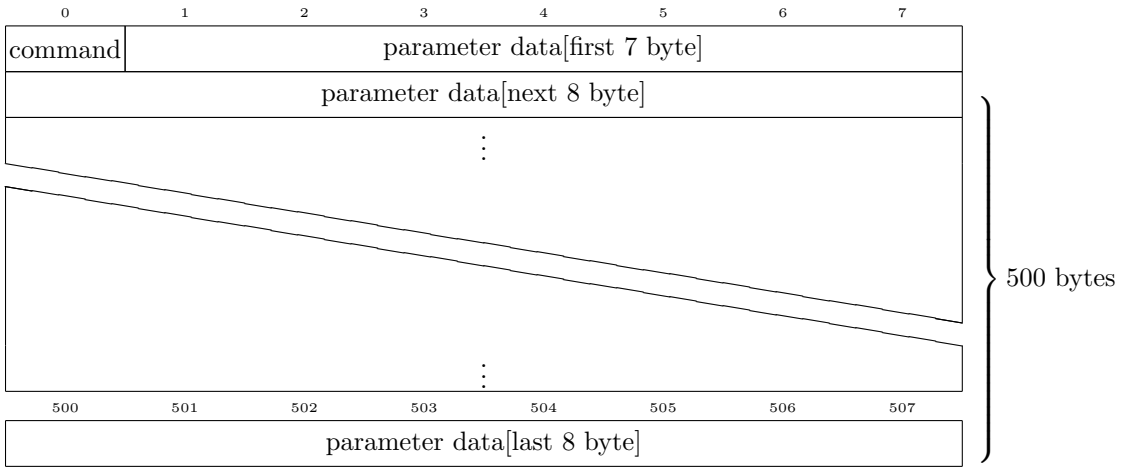
The remote control API of the PoolLab2 follows a Command-and-Response pattern, where each command sent to the PoolLab2 by a Bluetooth® Low Energy central device is acknowledged by a response with a **status-code** (or error-code) and, depending on the command, additional data.

4.2 Commands

`Commands` are sent to the PoolLab2 by writing the `command-code` followed by command-specific parameters or data into the **MOSI_CMD** `Characteristic`. The format is command-specific but always begins with the 1-byte `command-code`. The `Characteristic` is 508 bytes wide and unused bytes for a specific command shall be left `0`, and are not required to be updated (**partial writes are OK**). The application may use plain `write-without-response` or use the `write-with-response` scheme, both are supported. As the logical response is delivered by a separate mechanism anyway, the use of `write-without-response` is encouraged for performance reasons.

4.2.1 Command Layout

The commands are delivered by writing into the **MOSI_CMD** `Characteristic` and the data is formatted according to the figure below:



`Fig.1: Command packet layout`

A list of available commands is given in the next sections of this document.

4.3 Responses

Each `Command` is followed by a `Response`. For performance reasons, the `Response` mechanism is split into two `Characteristics`. When the PoolLab2 has finished processing a `Command` sent to it, the **MISO_SIG** `Characteristic` is updated with 8 bytes of data. A central-device application that has enabled `Notifications` of this `Characteristic` will receive these in an asynchronous manner, where the time between the `Command` and this `Notification` depends on the execution-time of the specific `Command`.

There are two notable exceptions to this scheme, the `RESET` and `SLEEP` commands. These generally do return a response, however, the PoolLab2 firmware will immediately enter reset or sleep-mode after transmission of this signal. Some host platforms tend to omit the **MISO_SIG** `Notification` as the underlying transport interface will also signal the `DEVICE_DISCONNECTED` signal with preference. This should be interpreted as successful command execution by the application.

When the central-device application sends any `Command`, it shall expect to receive a `Notification` of the **MISO_SIG** `Characteristic`. It shall then read the first 2 bytes of this `Characteristic`, the first being the **reponse-type** and the second the **status-code**, check the **status-code** equals `CMD_SUCCESS`, and then, if the **reponse-type** equals `TYPE_READMISO`, issue a read-request to the PoolLab2's **CMD_MISO** `Characteristic`.

4.3.1 Response Notification

The `Notification` data is 8 bytes wide and the general layout is given below:

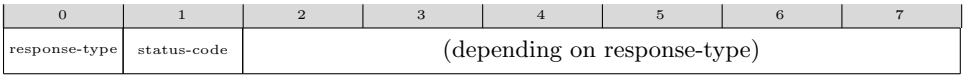


Fig.2: Response Notification layout

The first byte of the data received through the **MISO_SIG** `Notification` indicates the **reponse-type**, and the second byte indicates the **status-code**. The remaining 6 byte of the `Notification` data are used only if the **reponse-type** is `TYPE_EXTENDED` or `TYPE_READMISO`, and set to 0 otherwise.

4.3.2 Response Types

There are 3 possible response-types, and each has a different amount of data that is included in the **MISO_SIG** `Notification`. The types, the hexadecimal encoding, and the amount of usable bytes left in the **MISO_SIG** `Notification` after the **status-code** (the first 2 bytes) are given below:

Response-type	Hex-Value	Data Bytes
`TYPE_SIMPLE`	**0x40**	0
`TYPE_EXTENDED`	**0x41**	0-6
`TYPE_READMISO`	**0x42**	2

Table 2: **Response-types** returned by the PoolLab2's `Notification`

4.3.3 Response Type `TYPE_SIMPLE`

This type indicates that the **MISO_SIG** `Notification` contains only 2 useable bytes (see Fig.2 in section "Response Notification"), and that there is no additional data to be read by the application. The application shall process the **status-code** only.

The PoolLab2 generally uses this response scheme for commands that return only a status-code and no addtional data. An example is the `SET_DISP_BRIGHTNESS` command that changes the LCD's backlight intensity and will return either the `CMD_SUCCESS` code, or the error code indicating the requested brightness is outside of the parameter range (`CMD_ERR_PARAM`).

4.3.4 Response Type `TYPE_EXTENDED`

This type indicates that the **MISO_SIG** `Notification` contains up to 8 useable bytes, and that there is no additional data to be read by the application. The application shall process the **status-code** and additional bytes from the `Notification` data depending on the command used.

0	1	2	3	4	5	6	7
0x41	status-code	(depending on command)					

`Fig.3: TYPE_EXTENDED Notification layout`

An example is the `GET_BATTERY_VOLTAGE` command that returns the PoolLab2's last measured battery voltage in milliVolts, formatted as 4 byte little-endian 32-bit unsigned integer. After receiving this command, the PoolLab2 will update it's **MISO_SIG** `Notification` characteristic and notify the central device.
The `Notification` data will then be formatted according to Fig.4 below:

0	1	2	3	4	5	6	7
0x41	0x01	battery-mV				0x00	0x00

`Fig.4: GET_BATTERY_VOLTAGE Notification layout`

Here, 0x41 indicates the `TYPE_EXTENDED` Response Type, 0x01 indicates the command executed successfully (`CMD_SUCCESS`), and bytes 2, 3, 4 and 5 are the little-endian formatted 32-bit unsigned integer representation of the device's battery voltage (in mV). Bytes 6 and 7 are unused, and set to 0x00.

4.3.5 Response Type `TYPE_READMISO`

The **reponse-type** `TYPE_READMISO` indicates to the central-device application that the response-data of the given `Command` are available to be read by the central-device by sending a GATT `Read-Request` to the **MISO_CMD** `Characteristic`. In this case, the `Notification` data contains 4 byte of information: the **reponse-type**, the **status-code**, and 2 byte indicating the number of bytes available to be read inside the **MISO_CMD** `Characteristic`. The `Notification` data will be formatted according to Fig.5 below:

0	1	2	3	4	5	6	7
0x42	status-code	data-length		0x00	0x00	0x00	0x00

`Fig.5: TYPE_READMISO Notification layout`

The `data-length` is a little-endian 16-bit unsigned integer and indicates the amount of data available in the **MISO_CMD** `Characteristic`. This value will never exceed 508. The remaining 4 bytes (4, 5, 6, 7) are always set to 0x00. Receiving this response shall trigger the central-device application to read the **MISO_CMD** `Characteristic` and parse the data depending the on specific command used.

4.4 Status Codes

The following **Status-Codes** may be returned as the first byte of the **MISO_SIG** `Notification` data.

Status-Code	Hex-Value	Description
CMD_SUCCESS	**0x01**	Command executed successfully
CMD_ERR_UNKNOWN	**0x02**	Received unknown command-code
CMD_ERR_NOTAUTHORIZED	**0x03**	Device not in Factory-Test Mode
CMD_ERR_BATTERYLOW	**0x04**	Battery too low to execute given command
CMD_ERR_PARAM	**0x05**	Parameter count or range for the given command is invalid
CMD_ERR_DB_READONLY	**0x06**	The device database cannot be updated, a reset is required
CMD_ERR_ALREADY_ACTIVE	**0x40**	Cannot execute command, a firmware or database upgrade is started
CMD_ERR_NOT_ACTIVE	**0x41**	The given command requires the device to be in firmware- or database-upgrade mode
CMD_ERR_OTA	**0x42**	Firmware or database upgrade failed

Table 3: **Status-Codes** returned by the PoolLab2's **MISO_SIG** `Notification`

5 API Commands

5.1 Command list

Command	Hex-Value	Min. FW. Version
RESET	**0x01**	**1**
SLEEP	**0x02**	**1**
GET_BATTERY_VOLTAGE	**0x03**	**1**
GET_QUICK_INFO	**0x04**	**1**
GET_UPDATE_API_PATH	**0x06**	**2**
GET_CLOUD_API_PATH	**0x07**	**2**
SET_UPDATE_API_PATH	**0x08**	**2**
SET_CLOUD_API_PATH	**0x09**	**2**
GET_TIMEZONE	**0x0A**	**1**
GET_WIFI_SSID	**0x0B**	**1**
GET_SOURCES	**0x0C**	**1**
CLEAR_SOURCES	**0x0D**	**1**
ADD_SOURCE	**0x0E**	**1**
FLUSH_USERDATA	**0x0F**	**1**
SET_WIFI_LOGINS	**0x10**	**1**
SET_CLOUD_LOGIN	**0x11**	**1**
SET_DISP_BRIGHTNESS	**0x12**	**1**
SET_EPOCH_TIME	**0x13**	**1**
SET_TIMEZONE	**0x14**	**1**
SET_AUTO_DIM_AUTO_OFF	**0x15**	**4**
GET_MEASUREMENTS	**0x21**	**1**
CLEAR_MEASUREMENTS	**0x22**	**1**

Table 4: General commands related to device configuration and data synchronization

5.2 OTA Command List

Command	Hex-Value	Min. FW. Version
OTA_FWUPDATE_GET_STATE	**0x40**	**4**
OTA_FWUPDATE_START	**0x41**	**4**
OTA_FWUPDATE_WRITE	**0x42**	**4**
OTA_FWUPDATE_FINISH	**0x43**	**4**
OTA_DBUPDATE_GET_STATE	**0x50**	**6**
OTA_DBUPDATE_START	**0x51**	**6**
OTA_DBUPDATE_START_SEGMENT	**0x52**	**6**
OTA_DBUPDATE_WRITE	**0x53**	**6**
OTA_DBUPDATE_FINISH_SEGMENT	**0x54**	**6**
OTA_DBUPDATE_FINISH	**0x55**	**6**
OTA_DBUPDATE__RESERVED_1	**0x56**	**-**
OTA_DBUPDATE_FINISH_TO_RODATA	**0x57**	**6**
OTA_DBUPDATE__RESERVED_2	**0x58**	**-**
OTA_DBUPDATE__RESERVED_3	**0x59**	**-**

Table 5: Commands related to database and device firmware upgrade API

5.3 RESET Command

Immediately causes the PoolLab2 to reset it's microcontroller. A success-response is sent, but may be lost since the underlying transport layer will be disconnected immediately.

Command: RESET
Hex Value: 0x01
Parameters: -
Minimum FW Version: 1

5.3.1 Input

Example input into the **MOSI_CMD** Characteristic.

0	1	2	3	4	5	6	7
0x01	-	-	-	-	-	-	-

5.3.2 Response

The Response will be delivered by the **MISO_SIG** Notification and the **response-type** will always be TYPE_SIMPLE.

0	1	2	3	4	5	6	7
0x40	status-code	-	-	-	-	-	-

This command will never return an error (the **status-code** will always be CMD_SUCCESS).

5.4 SLEEP Command

Immediately causes the PoolLab2 to enter sleep-mode. A success-response is sent, but may be lost since the underlying transport layer will be disconnected immediately.

Command: SLEEP
Hex Value: 0x02
Parameters: -
Minimum FW Version: 1

5.4.1 Input

Example input into the **MOSI_CMD** Characteristic.

0	1	2	3	4	5	6	7
0x02	-	-	-	-	-	-	-

5.4.2 Response

The Response will be delivered by the **MISO_SIG** Notification and the **response-type** will always be TYPE_SIMPLE.

0	1	2	3	4	5	6	7
0x40	status-code	-	-	-	-	-	-

This command will never return an error (the **status-code** will always be CMD_SUCCESS).

5.5 GET_BATTERY_VOLTAGE Command

Returns the latest battery voltage measurement of the PoolLab2, in millivolts. The PoolLab2 constantly monitors it's battery voltage so the latest reading is usually never older than a few seconds. The expected output range is 3550 to 4600 mV. Database and device firmware upgrade operations should not be attempted if the battery voltage is below 3850 millivolts.

Command: `GET_BATTERY_VOLTAGE`
Hex Value: `0x03`
Parameters: -
Minimum FW Version: `1`

5.5.1 Input

Example input into the **MOSI_CMD** `Characteristic`.

0	1	2	3	4	5	6	7
0x03	-	-	-	-	-	-	-

5.5.2 Response

The `Response` will be delivered by the **MISO_SIG** `Notification` and the **response-type** will be `TYPE_EXTENDED` unless an error occurrs.

0	1	2	3	4	5	6	7
0x41	status-code	battery-mV				0x00	0x00

Bytes 2, 3, 4 and 5 are the little-endian formatted 32-bit unsigned integer representation of the device's battery voltage (in mV). Bytes 6 and 7 are unused, and set to 0x00.

5.6 GET_QUICK_INFO Command

Returns a collection of most of the PoolLab2's device information and configuration data.

Command: GET_QUICK_INFO
Hex Value: 0x04
Parameters: -
Minimum FW Version: 1

5.6.1 Input

Example input into the **MOSI_CMD** Characteristic.

0	1	2	3	4	5	6	7
0x04	-	-	-	-	-	-	-

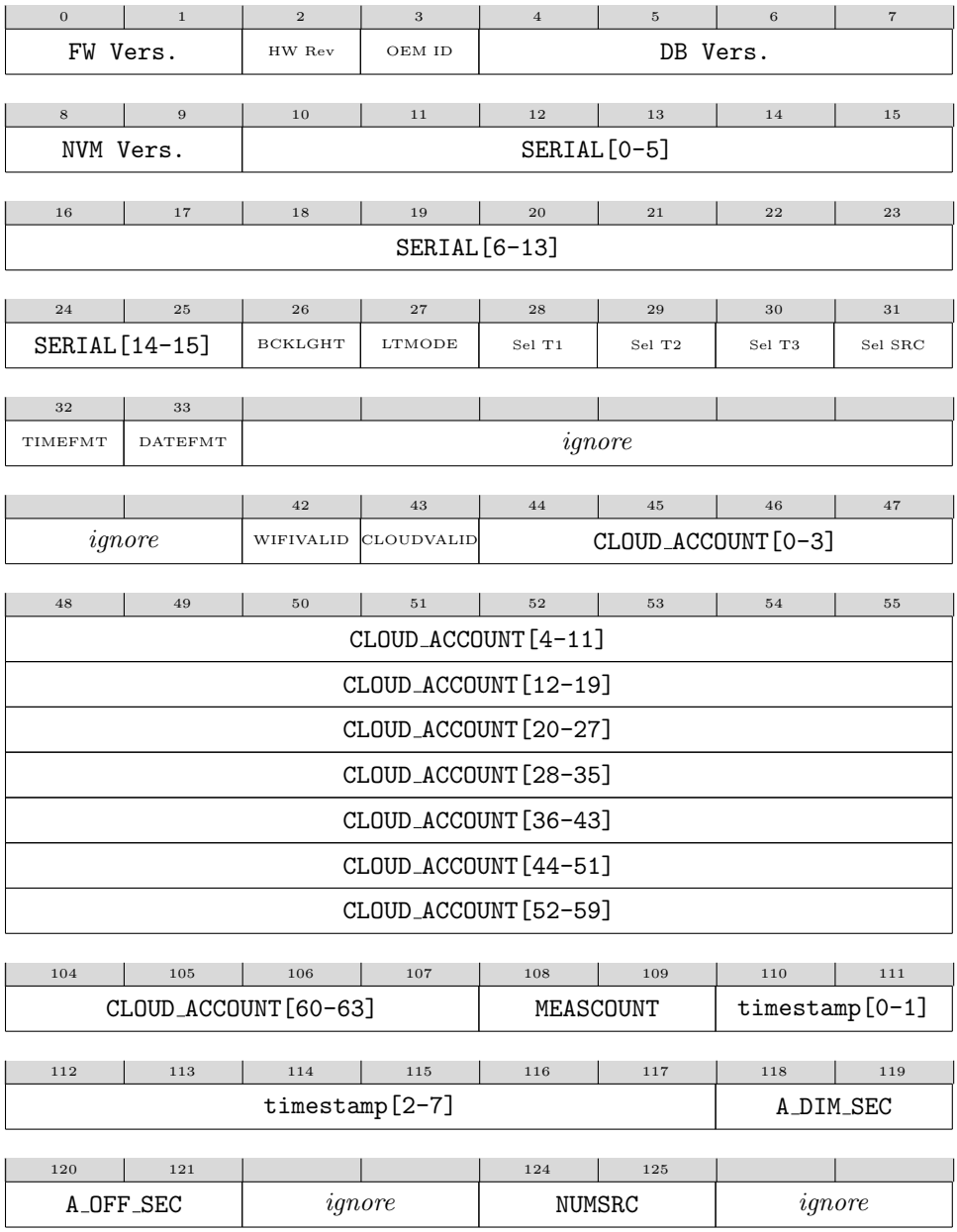
5.6.2 Response

The Response will be delivered by the **MISO_CMD** Characteristic. The PoolLab2 will first emit a **MISO_SIG** Notification and the **response-type** will be TYPE_READMISO unless an error occurrs.

0	1	2	3	4	5	6	7
0x42	status-code	read-length		0x00	0x00	0x00	0x00

Bytes 2 and 3 are the 16-bit unsigned integer representation of the read-length, which is the amount of bytes available to be read by the central device from the PoolLab2's **MISO_CMD** Characteristic.

Unless an error occurs, the `read-length` will be 128, and the **MISO_CMD** `Characteristic` has the following layout:



5.6.3 Fields

`[FW Vers.]` is the firmware version, as 16-bit little-endian integer.

`[HW Rev]` is the hardware revision, as 8-bit integer. Currently, there is only one revision (this value will be 0x01).

`[OEM ID]` is the device OEM ID, as 8-bit integer. See the OEM section for more information.

`[DB Vers.]` is the currently installed database version, as 32-bit little-endian integer.

`[NVM Vers.]` this value can be ignored.

[SERIAL] is the PoolLab2's serial number, encoded as 16-character ASCII string.

[BCKLGHT] is the currently set backlight-level of the PoolLab2's LCD, as 8-bit integer. The valid range is 0 to 15.

[LTMODE] is set to 0x01 if the user has switched the PoolLab2 into liquid-measurement-mode, otherwise it will be set to 0x00.

[Sel T1] is an 8-bit integer representing the user-selected menu-index of the measurement chamber 1.

[Sel T2] is an 8-bit integer representing the user-selected menu-index of the measurement chamber 2.

[Sel T3] is an 8-bit integer representing the user-selected menu-index of the measurement chamber 3.

[TIMEFMT] is set to 0x00 if the user-selected time-format is the 12-hour-format ("AM/PM"), and set to 0x01 if the user selected the 24-hour-format.

[DATEFMT] is set to 0x00 if the user-selected date-format is DD.MM.YYYY, and set to 0x01 if the user selected the MM.DD.YYYY format.

[WIFIVALID] is set to 0x01 if the PoolLab2 has saved WiFi access point login data.

[CLOUDVALID] is set to 0x01 if the PoolLab2 has saved valid Cloud login data.

[CLOUD_ACCOUNT] is a 64-character ASCII string representing the Cloud login account name. Note the *password* cannot be retrieved.

[MEASCOUNT] is a 16-bit little-endian integer that indicates the number of saved measurement data on the PoolLab2.

[TIMESTAMP] is a 64-bit little-endian integer that represents the PoolLab2's clock-time. It is an *Epoch Timestamp*, that counts the elapsed seconds since 1/1/1970 00:00:00 UTC.

[A_DIM_SEC] is a 16-bit little-endian integer that indicates after how many seconds of inactivity the PoolLab2 will automatically dim the backlight of it's LCD.

[A_OFF_SEC] is a 16-bit little-endian integer that indicates after how many seconds of inactivity the PoolLab2 will automatically enter sleep-mode. Note that the PoolLab2 will never automatically enter sleep-mode while a Bluetooth® Low Energy central device is connected, unless the battery-level is critical.

[NUMSRC] is a 16-bit little-endian integer that represents the number of Sources saved on the device. Sources are customizable and user-selectable names that are used to organize measurement data. The limit is 20.

5.7 GET_UPDATE_API_PATH Command

Returns the currently configured `Update path`. This is an ASCII-encoded URL (without 'https://' prefix) of up to 128 characters length. This is the base-url that is used by the firmware to check for automatic firmware and database upgrades using the WiFi connection. For more information, see the Update Section.

Command: `GET_UPDATE_API_PATH`
Hex Value: `0x06`
Parameters: -
Minimum FW Version: `2`

5.7.1 Input

Example input into the **MOSI_CMD** `Characteristic`.

0	1	2	3	4	5	6	7
0x06	-	-	-	-	-	-	-

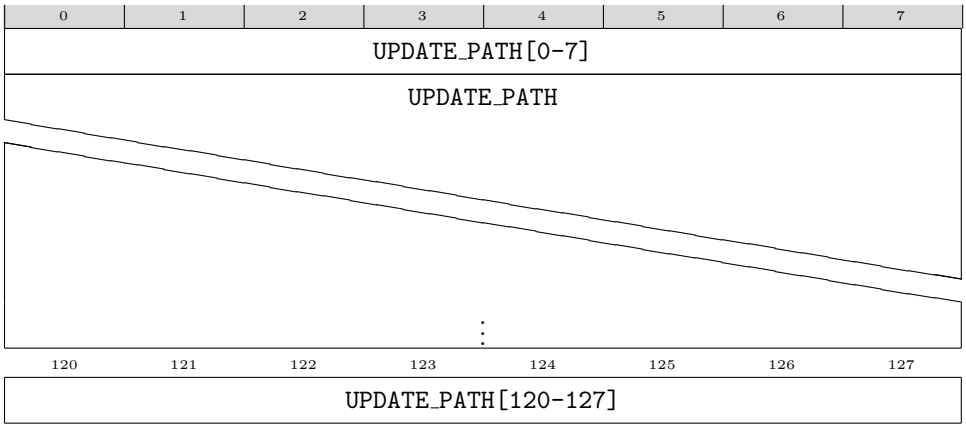
5.7.2 Response

The `Response` will be delivered by the **MISO_CMD** `Characteristic`. The PoolLab2 will first emit a **MISO_SIG** `Notification` and the **response-type** will be `TYPE_READMISO` unless an error occurrs.

0	1	2	3	4	5	6	7
0x42	status-code	read-length		0x00	0x00	0x00	0x00

Bytes 2 and 3 are the 16-bit unsigned integer representation of the `read-length`, which is the amount of bytes available to be read by the central device from the PoolLab2's **MISO_CMD** `Characteristic`.

Unless an error occurs, the `read-length` will be up to 128, and the **MISO_CMD** `Characteristic` has the following layout:



Unused characters are set to `0x00`, however the application should not assume this value to be 0-terminated: in case the value is actually 128 characters long, no 0-terminator is included.

5.8 GET_CLOUD_API_PATH Command

Returns the currently configured `Cloud Server URL`. This is an ASCII-encoded URL (without 'https://' prefix) of up to 128 characters length. This is the base-url that is used to communcation with the Cloud application endpoint over the WiFi connection. For more information, see the Cloud Section.

Command: `GET_CLOUD_API_PATH`
Hex Value: `0x07`
Parameters: -
Minimum FW Version: `2`

5.8.1 Input

Example input into the **MOSI_CMD** `Characteristic`.

0	1	2	3	4	5	6	7
0x07	-	-	-	-	-	-	-

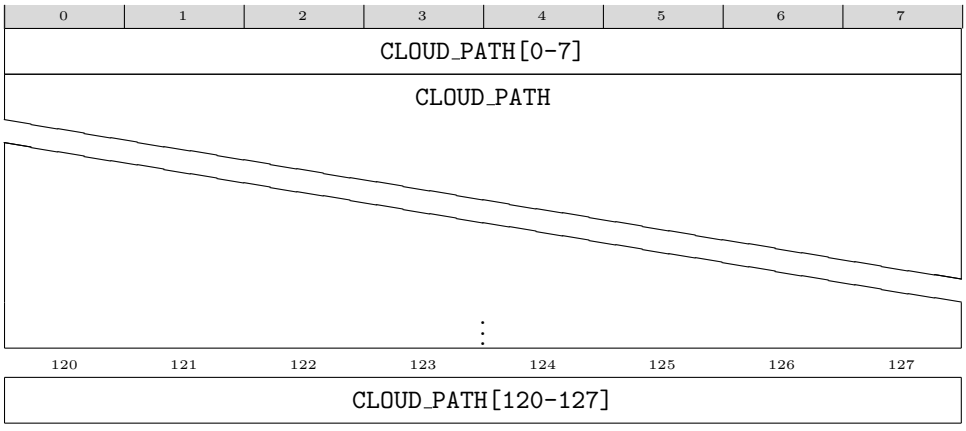
5.8.2 Response

The `Response` will be delivered by the **MISO_CMD** `Characteristic`. The PoolLab2 will first emit a **MISO_SIG** `Notification` and the **response-type** will be `TYPE_READMISO` unless an error occurrs.

0	1	2	3	4	5	6	7
0x42	status-code	read-length		0x00	0x00	0x00	0x00

Bytes 2 and 3 are the 16-bit unsigned integer representation of the `read-length`, which is the amount of bytes available to be read by the central device from the PoolLab2's **MISO_CMD** `Characteristic`.

Unless an error occurs, the `read-length` will be up to 128, and the **MISO_CMD** `Characteristic` has the following layout:



Unused characters are set to `0x00`, however the application should not assume this value to be 0-terminated: in case the value is actually 128 characters long, no 0-terminator is included.

5.9 SET_UPDATE_API_PATH Command

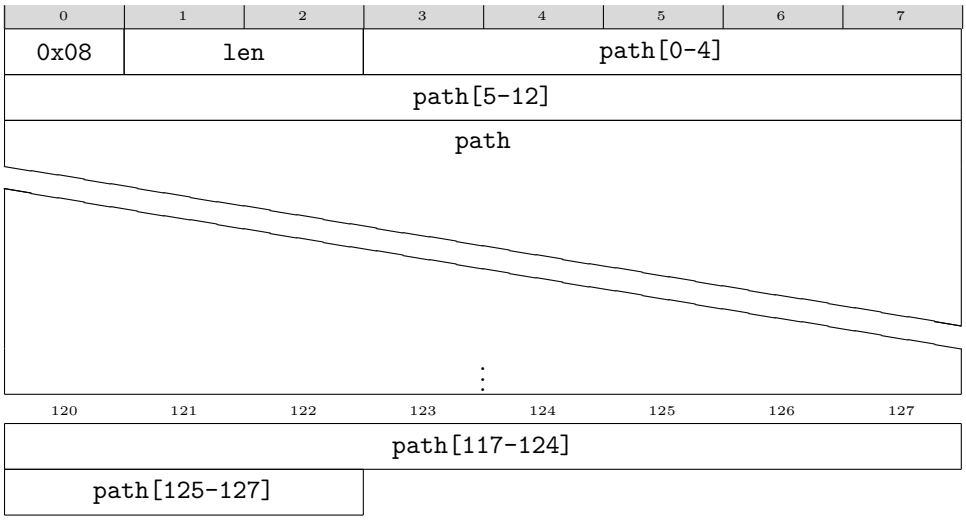
Sets the `Update path`. This must be an ASCII-encoded URL (without 'https://' prefix) of up to 128 characters length. This is the base-url that is used by the firmware to check for automatic firmware and database upgrades using the WiFi connection. For more information, see the Update Section.

Command: `SET_UPDATE_API_PATH`
Hex Value: `0x08`
Parameters: `len`, `path`
Minimum FW Version: `2`

5.9.1 Input

The first parameter, `len`, is a 16-bit unsigned integer that is the length of the `path` parameter. Currently, this value MUST be set to 128.

The second parameter, `data`, is the new `Update path` to be set. This must be ASCII-encoded, and less or equal to 128 characters in length. Unused characters are then to be filled with `0x00`. This means the total command length for this command is always 131 bytes: 1 byte Command Code, 2 byte for the `len` parameter (set to 128), and 128 characters for the URL.



5.9.2 Response

The `Response` will be delivered by the **MISO_SIG** `Notification` and the **response-type** will be `TYPE_SIMPLE`.

0	1	2	3	4	5	6	7
0x40	status-code	-	-	-	-	-	-

The changes are effective immediately.

5.10 SET_CLOUD_API_PATH Command

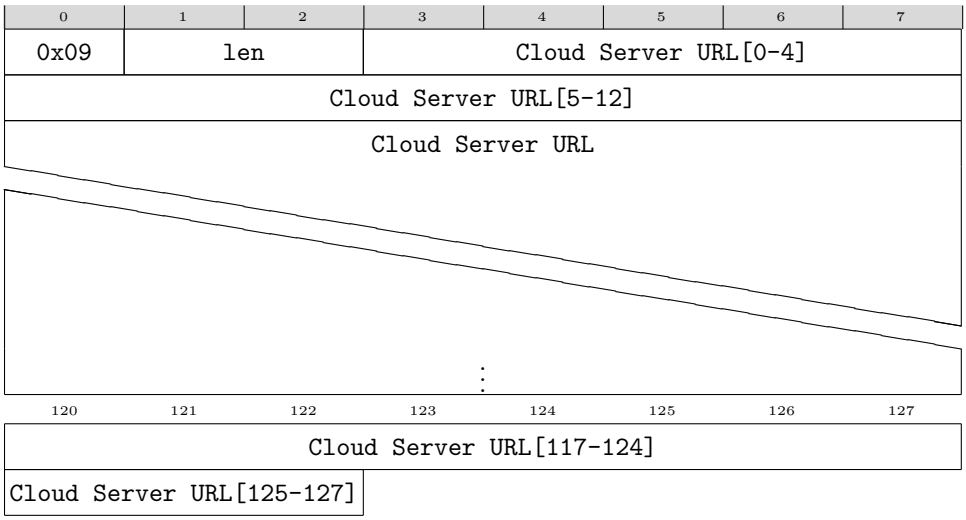
Sets the Cloud Server URL. This must be an ASCII-encoded URL (without 'https://' prefix) of up to 128 characters length. This is the base-url that is used to communcation with the Cloud application endpoint over the WiFi connection. For more information, see the Cloud Section.

Command: SET_CLOUD_API_PATH
Hex Value: 0x09
Parameters: len, path
Minimum FW Version: 2

5.10.1 Input

The first parameter, len, is a 16-bit unsigned integer that is the length of the path parameter. Currently, this value MUST be set to 128.

The second parameter, data, is the new Cloud Server URL to be set. This must be ASCII-encoded, and less or equal to 128 characters in length. Unused characters are then to be filled with 0x00. This means the total command length for this command is always 131 bytes: 1 byte Command Code, 2 byte for the len parameter (set to 128), and 128 characters for the URL.



5.10.2 Response

The Response will be delivered by the **MISO_SIG** Notification and the **response-type** will be TYPE_SIMPLE.

0	1	2	3	4	5	6	7
0x40	status-code	-	-	-	-	-	-

The changes are effective immediately.

5.11 GET_TIMEZONE Command

Returns the currently configured `Timezone`. This is an ASCII-encoded string representing the **POSIX Timezone String**.

Command: `GET_TIMEZONE`
Hex Value: `0x0A`
Parameters: -
Minimum FW Version: `1`

5.11.1 Input

Example input into the **MOSI_CMD** `Characteristic`.

0	1	2	3	4	5	6	7
0x0A	-	-	-	-	-	-	-

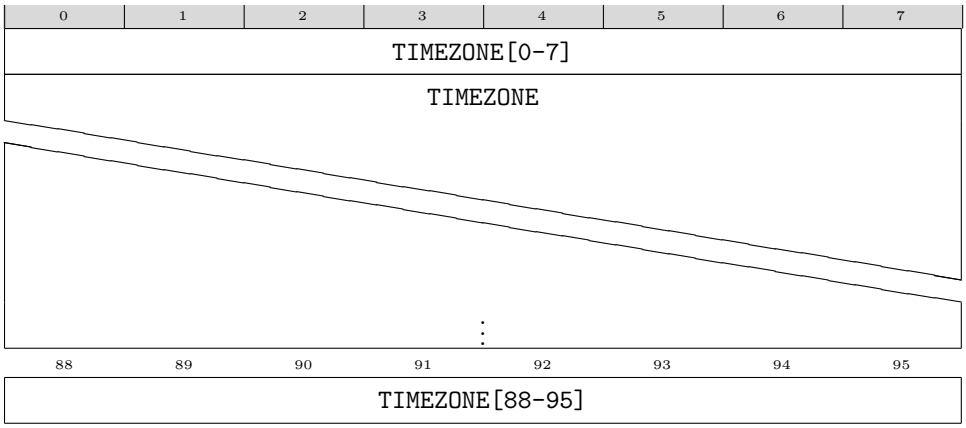
5.11.2 Response

The `Response` will be delivered by the **MISO_CMD** `Characteristic`. The PoolLab2 will first emit a **MISO_SIG** `Notification` and the **response-type** will be `TYPE_READMISO` unless an error occurrs.

0	1	2	3	4	5	6	7
0x42	status-code	read-length		0x00	0x00	0x00	0x00

Bytes 2 and 3 are the 16-bit unsigned integer representation of the `read-length`, which is the amount of bytes available to be read by the central device from the PoolLab2's **MISO_CMD** `Characteristic`.

Unless an error occurs, the `read-length` will be up to 96, and the **MISO_CMD** `Characteristic` has the following layout:



Unused characters are set to `0x00`, however the application should not assume this value to be 0-terminated: in case the value is actually 96 characters long, no 0-terminator is included.

Note the timezone returned by the device can be up to 96 characters long, however when *setting* the timezone via the `SET_TIMEZONE` command only up to 64 characters are allowed. This is because the software stack in the firmware can potentially expand the string to match it's database.

5.12 GET_WIFI_SSID Command

Returns the currently configured `WiFi Access Point SSID`. This is an ASCII-encoded string.

Command: `GET_WIFI_SSID`
Hex Value: `0x0B`
Parameters: -
Minimum FW Version: `1`

5.12.1 Input

Example input into the **MOSI_CMD** `Characteristic`.

0	1	2	3	4	5	6	7
0x0B	-	-	-	-	-	-	-

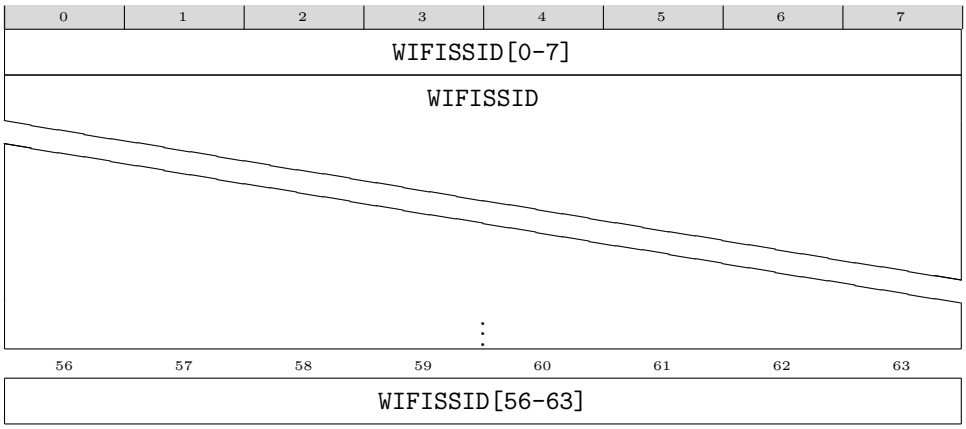
5.12.2 Response

The `Response` will be delivered by the **MISO_CMD** `Characteristic`. The PoolLab2 will first emit a **MISO_SIG** `Notification` and the **response-type** will be `TYPE_READMISO` unless an error occurrs.

0	1	2	3	4	5	6	7
0x42	status-code	read-length		0x00	0x00	0x00	0x00

Bytes 2 and 3 are the 16-bit unsigned integer representation of the `read-length`, which is the amount of bytes available to be read by the central device from the PoolLab2's **MISO_CMD** `Characteristic`.

Unless an error occurs, the `read-length` will be up to 64, and the **MISO_CMD** `Characteristic` has the following layout:



Unused characters are set to `0x00`, however the application should not assume this value to be 0-terminated: in case the value is actually 64 characters long, no 0-terminator is included.

5.13 GET_SOURCES Command

Returns the requested part of the list of Sources currently configured on the PoolLab2.

The Sources are stored in a compacted memory format that is directly accessed by this command. Since the total memory occupied by the Source database exceeds the maximum amount of data that can be transmitted with a single Response, this command expects an offset (into the PoolLab2's Source database), and the application is expected to issue as many GET_SOURCES commands as necessary to retrieve a complete list of all Sources.

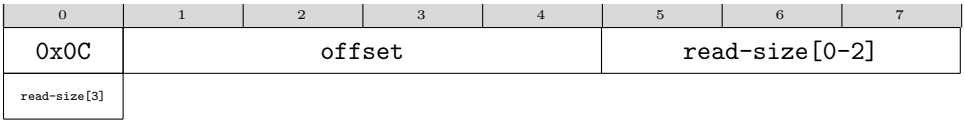
Command: GET_SOURCES
Hex Value: 0x0C
Parameters: offset, read-size
Minimum FW Version: 1

5.13.1 Input

The offset is a 32-bit little-endian unsigned integer. It is used to specify the start of the range of data to be returned with the Response. One Source occupies 60 bytes of memory, and there can be at maximum 20 saved Sources, thus the valid range is 0 to 1199.

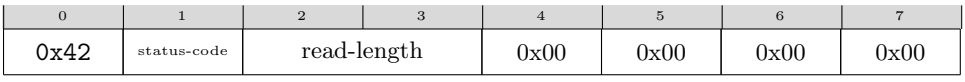
The read-size is a 32-bit little-endian unsigned integer specifying the amount of Source data to be returned with the Response. The valid range is 1 to 480. Since the total memory space of the Source database is 1200 bytes, the offset and read-size parameters combined must not exceed 1200.

Command layout:



5.13.2 Response

The Response will be delivered by the **MISO_CMD** Characteristic. The PoolLab2 will first emit a **MISO_SIG** Notification and the **response-type** will be TYPE_READMISO unless an error occurrs.

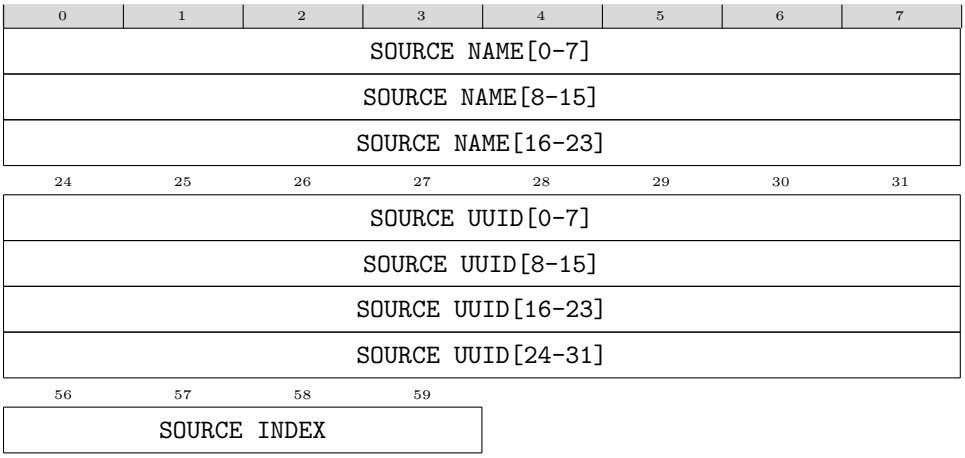


Bytes 2 and 3 are the 16-bit unsigned integer representation of the read-length, which is the amount of bytes available to be read by the central device from the PoolLab2's **MISO_CMD** Characteristic.

Unless an error occurs, the read-length will be the same as the read-size input-parameter, and the **MISO_CMD** Characteristic contains a subset of the memory Source database, at the requested range.

Sources are 60 bytes long data structures. They are made up of a Source Name, which is a 24-character ASCII string that is displayed on the PoolLab2 LCD, a 32-byte UUID, and a 4-byte unsigned integer index. The index is used to cross-reference measurement data records.

The memory layout of a Source is given below, the Source database is an array of 20 of these structures.



A simple way to read all available `Sources` would be to first determine the amount of `Sources` saved on the device by using the `GET_QUICK_INFO` command, and then to execute as many `GET_SOURCES` commands with a constant `read-size` of 60, and the `offset` parameter increasing by 60 with each subsequent `GET_SOURCES` command.

Since the `GET_SOURCES` command supports `read-size`s up to 480 byte, up to 8 `Sources` can be read with a single `GET_SOURCES` command.

5.14 CLEAR_SOURCES Command

Deletes all configured Sources from the PoolLab2. If no Sources are added before the connection is closed, a default Source will be created on the device automatically.

Command: CLEAR_SOURCES
Hex Value: 0x0D
Parameters: -
Minimum FW Version: 1

5.14.1 Input

Example input into the **MOSI_CMD** Characteristic.

0	1	2	3	4	5	6	7
0x0D	-	-	-	-	-	-	-

5.14.2 Response

The Response will be delivered by the **MISO_SIG** Notification and the **response-type** will always be TYPE_SIMPLE.

0	1	2	3	4	5	6	7
0x40	status-code	-	-	-	-	-	-

Note all measurement data are lost when this command is executed.

Note that the FLUSH_USERDATA command must be executed after this command, otherwise the changes will be lost on device-reset.

5.15 ADD_SOURCE Command

Adds a `Source` to the list of `Sources` on the PoolLab2.

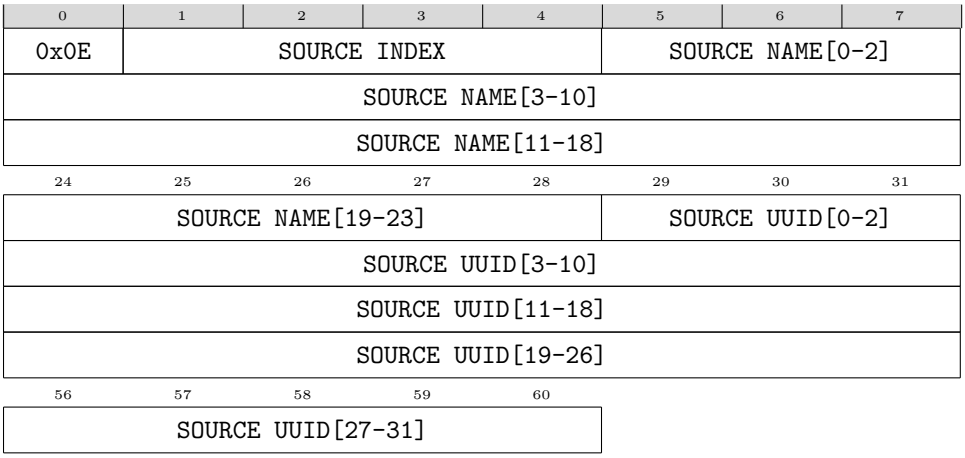
Command: `ADD_SOURCE`
Hex Value: `0x0E`
Parameters: `source-index`, `source-name`, `source-uuid`
Minimum FW Version: `1`

5.15.1 Input

The `source-index` parameter is a 32-bit unsigned integer that will be saved to `Measurement` records when the given `Source` is selected and a measurement is taken.

The `source-name` parameter is a 24-character ASCII string that is shown as the name of the `Source` on the PoolLab2's main menu. It must contain only ASCII-Characters.

The `source-uuid` parameter is a 32-byte value that is used to cross-reference this `Source` with the Cloud Server API.



5.15.2 Response

The Response will be delivered by the **MISO_SIG** Notification and the **response-type** will always be TYPE_SIMPLE.

0	1	2	3	4	5	6	7
0x40	status-code	-	-	-	-	-	-

Note that the FLUSH_USERDATA command must be executed after all desired Sources have been added to the PoolLab2, otherwise the changes will be lost on device-reset.

5.16 FLUSH_USERDATA Command

Writes changes to the Source database to the PoolLab2's persistant memory. This command can take up to 10 seconds to execute.

Command: FLUSH_USERDATA
Hex Value: 0x0F
Parameters: -
Minimum FW Version: 1

5.16.1 Input

Example input into the **MOSI_CMD** Characteristic.

0	1	2	3	4	5	6	7
0x0F	-	-	-	-	-	-	-

5.16.2 Response

The Response will be delivered by the **MISO_SIG** Notification and the **response-type** will always be TYPE_SIMPLE.

0	1	2	3	4	5	6	7
0x40	status-code	-	-	-	-	-	-

5.17 SET_WIFI_LOGINS Command

Sets the `WiFi Access Point Logins`. The SSID and password must be ASCII-encoded. Certificate-based authentication available on request.

Command: `SET_WIFI_LOGINS`
Hex Value: `0x10`
Parameters: `len-ssid`, `len-pass`, `AP_SSID`, `AP_PASSWD`
Minimum FW Version: `1`

5.17.1 Input

The first parameter, `len-ssid`, is a 16-bit unsigned integer that is the length of the `AP_SSID` parameter. Valid range is 1 to 64 (inclusive).

The second parameter, `len-pass`, is a 16-bit unsigned integer that is the length of the `AP_PASSWD` parameter. Valid range is 0 to 32 (inclusive).

The third parameter, `AP_SSID`, is the `WiFi Acces Point SSID` to be set. This must be ASCII-encoded, and less or equal to 64 characters in length. The starting position for this parameter is always at byte-index 5 (first the command-byte, then two byte for both length parameters).

The last parameter, `AP_PASSWD`, is the `WiFi Acces Point Password` to be set. This must be ASCII-encoded, and less or equal to 32 characters in length. The starting position for this parameter is at byte-index [5 plus `len-ssid`].

It's possible to use this command with constant-length parameters, i.e. `len-ssid` set to 64 and `len-pass` set to 32, even when the values are shorter, by setting unused bytes for the last two parameters to **0x00**. This can be used to simplify application logic.

For connecting to unprotected (open) WiFi networks, use this command with `len-pass` set to 0.

To delete saved WiFi Logins from the PoolLab2, simply execute this command with both length-parameters set to 0.

The input data layout for a maximum-length AP_SSID and maximum-length AP_PASSWD is:

0	1	2	3	4	5	6	7
0x10	len-ssid		len-pass		AP_SSID		

8	9	10	11	12	13	14	15
AP_SSID							
AP_SSID							
AP_SSID							
AP_SSID							
AP_SSID							
AP_SSID							
AP_SSID							

64	65	66	67	68	69	70	71
AP_SSID					AP_PASSWD		
AP_PASSWD							
AP_PASSWD							
AP_PASSWD							

96	97	98	99	100			
AP_PASSWD					-		

5.17.2 Response

The Response will be delivered by the **MISO_SIG** Notification and the **response-type** will be TYPE_SIMPLE.

0	1	2	3	4	5	6	7
0x40	status-code	-	-	-	-	-	-

The changes are effective immediately.

5.18 SET_CLOUD_LOGIN Command

Saves login-information (username and password) to use when the PoolLab2 synchronizes it's measurements over WiFi with a `Cloud Server`.

Command: `SET_CLOUD_LOGINS`
Hex Value: `0x11`
Parameters: `len-user`, `len-pass`, `USERNAME`, `PASSWD`
Minimum FW Version: `1`

5.18.1 Input

The first parameter, `len-user`, is a 16-bit unsigned integer that is the length of the `USERNAME` parameter. Valid range is 1 to 64 (inclusive).

The second parameter, `len-pass`, is a 16-bit unsigned integer that is the length of the `PASSWD` parameter. Valid range is 0 to 64 (inclusive).

The third parameter, `USERNAME`, is the `Cloud Login Username` to be set. This must be ASCII-encoded, and less or equal to 64 characters in length. The starting position for this parameter is always at byte-index 5 (first the command-byte, then two byte for both length parameters).

The last parameter, `PASSWD`, is the `Cloud Login Password` to be set. This must be ASCII-encoded, and less or equal to 64 characters in length. The starting position for this parameter is at byte-index [5 plus `len-user`].

It's possible to use this command with constant-length parameters, i.e. `len-user` set to 64 and `len-pass` set to 64, even when the values are shorter, by setting unused bytes for the last two parameters to **0x00**. This can be used to simplify application logic.

To delete saved Cloud Logins from the PoolLab2, simply execute this command with both length-parameters set to 0.

The input data layout for a maximum-length USERNAME and maximum-length PASSWD is:

0	1	2	3	4	5	6	7
0x11	len-user		len-pass		USERNAME		

8	9	10	11	12	13	14	15
USERNAME							
USERNAME							
USERNAME							
USERNAME							
USERNAME							
USERNAME							
USERNAME							

64	65	66	67	68	69	70	71
USERNAME					PASSWD		
PASSWD							
PASSWD							
PASSWD							
PASSWD							
PASSWD							
PASSWD							
PASSWD							

132	133	134	135	136	137	138	139
PASSWD					-		

5.18.2 Response

The `Response` will be delivered by the **MISO_SIG** `Notification` and the **response-type** will be `TYPE_SIMPLE`.

0	1	2	3	4	5	6	7
0x40	status-code	-	-	-	-	-	-

The changes are effective immediately.

5.19 SET_DISP_BRIGHTNESS Command

Sets the PoolLab2's LCD Backlight brightness level. There are 17 levels (0 to 16), where 16 is the brightest and 0 means the backlight is turned OFF. The user can choose the backlight brightness using the on-device menu, here 4 steps are selectable (25%/50%/75%/100%), these correspond to the levels 4, 8, 12 and 16 available by this command. The backlight-level is applied immediately and will be saved and restored over device restarts. The central-device application can choose to completely turn off the PoolLab2's LCD backlight. The LCD will then be unreadable, but power-consumption is minimized. This could be desirable for in-line applications. However, to avoid problems with PoolLab2 devices becoming unusable, the backlight-level is restored to 8 when the device reboots or wakes up from deep-sleep with a backlight-level set to 0.
The PoolLab2's internal automatic backlight-dimming sets the backlight-level to 8 after `A_DIM_SEC` of inactivity. This feature is disabled when the backlight is set to level 8 or below.

Command: `SET_DISP_BRIGHTNESS`
Hex Value: `0x12`
Parameters: `brightness`
Minimum FW Version: `1`

5.19.1 Input

The parameter `brightness` is a 8-bit unsigned integer that is the new LCD display brightness to set. This value must be between 0 and 16 (inclusive).

0	1	2	3	4	5	6	7
0x12	brightness	-					

5.19.2 Response

The `Response` will be delivered by the **MISO_SIG** `Notification` and the **response-type** will be `TYPE_SIMPLE`.

0	1	2	3	4	5	6	7
0x40	status-code	-	-	-	-	-	-

The changes are effective immediately and saved over device-resets.

5.20 SET_EPOCH_TIME Command

Changes the on-device time by setting a new Epoch Timestamp.

Command: SET_EPOCH_TIME
Hex Value: 0x13
Parameters: timestamp
Minimum FW Version: 1

5.20.1 Input

The parameter timestamp is a 64-bit unsigned integer that is the new EPOCH Timestamp to use (elapsed seconds since 1/1/1970 UTC).

0	1	2	3	4	5	6	7
0x13	timestamp						
timestamp	-						

5.20.2 Response

The Response will be delivered by the **MISO_SIG** Notification and the **response-type** will be TYPE_SIMPLE.

0	1	2	3	4	5	6	7
0x40	status-code	-	-	-	-	-	-

The changes are effective immediately.

5.21 SET_TIMEZONE Command

Sets the timezone for the PoolLab2. This is used to convert the Epoch `timestamp` to wall-clock date and time.

Command: `SET_TIMEZONE`
Hex Value: `0x14`
Parameters: `len`, `timezone`
Minimum FW Version: `2`

5.21.1 Input

The first parameter, `len`, is a 16-bit unsigned integer that is the length of the `timezone` parameter. Valid range is 1 to 64 (inclusive).

The second parameter, `timezone`, is the new `Timezone` to be set and used on the PoolLab2. This must be ASCII-encoded, and less or equal to 64 characters in length. The supplied length can be set to a constant 64, unused bytes after the data are then to be filled with `0x00`.

The given `timezone` string will be matched against a list of known timezones from the IANA Timezone database. These generally correspond to the timezones used in any POSIX-compatible system (i.e. Linux).

The IANA database can be found online: `https://www.iana.org/time-zones`

The input data layout for a maximum-length `timezone` is:

0	1	2	3	4	5	6	7
0x14	len		timezone				
timezone							
timezone							
timezone							
timezone							
timezone							
timezone							
timezone							
timezone			-				

5.21.2 Response

The `Response` will be delivered by the **MISO_SIG** `Notification` and the **response-type** will be `TYPE_SIMPLE`.

0	1	2	3	4	5	6	7
0x40	status-code	-	-	-	-	-	-

The changes are effective immediately.

5.22 SET_AUTO_DIM_AUTO_OFF Command

Sets the amount of seconds of inactivity after which the PoolLab2 will dim it's backlight or go into sleep-mode.

Command: SET_TIMEZONE
Hex Value: 0x15
Parameters: A_DIM_SEC, A_OFF_SEC
Minimum FW Version: 4

5.22.1 Input

The first parameter, A_DIM_SEC, is 32-bit little-endian unsigned integer that sets the amount of seconds of inactivity after which the PoolLab2 will automatically dim it's backlight. A value of 0 disables the auto-dimming feature. A value greater than is set for the the auto-sleep feature (the next parameter) also effectively disables this feature. Valid range for this parameter is 0 to 65535 (inclusive).

The second parameter, A_OFF_SEC, is 32-bit little-endian unsigned integer that sets the amount of seconds of inactivity after which the PoolLab2 will automatically go into sleep-mode. Valid range for this parameter is 0, and 120 to 65535 (inclusive), values between 0 and 120 are rejected. A value of 0 disables the auto-sleep feature.

To prolong battery-life of the PoolLab2, it is recommended to leave both features enabled.

The input data layout is:

0	1	2	3	4	5	6	7
0x15	A_DIM_SEC				A_OFF_SEC		
A_OFF_SEC	-						

5.22.2 Response

The Response will be delivered by the **MISO_SIG** Notification and the **response-type** will be TYPE_SIMPLE.

0	1	2	3	4	5	6	7
0x40	status-code	-	-	-	-	-	-

The changes are effective immediately.

5.23 GET_MEASUREMENTS Command

Requests are subset of the measurement-record memory from the PoolLab2. This commands works analog to the GET_SOURCES command.

The Measurements are stored in a compacted memory format that is directly accessed by this command. Since the total memory occupied by the Measurement database exceeds the maximum amount of data that can be transmitted with a single Response, this command expects an offset (into the PoolLab2's Memory database), and the application is expected to issue as many GET_MEASUREMENT commands as necessary to retrieve a complete list of all Measurements.

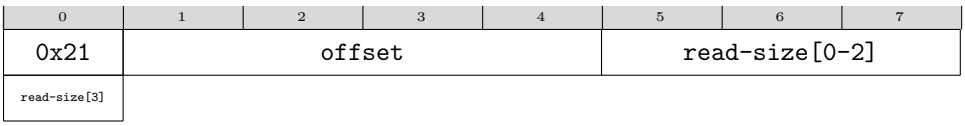
Command: GET_MEASUREMENTS
Hex Value: 0x21
Parameters: offset, read-size
Minimum FW Version: 1

5.23.1 Input

The offset is a 32-bit little-endian unsigned integer. It is used to specify the start of the range of data to be returned with the Response. One Measurement occupies 24 bytes of memory, and there can be at maximum 1024 saved Measurements, thus the valid range is 0 to 24576.

The read-size is a 32-bit little-endian unsigned integer specifying the amount of Measurement data (in bytes) to be returned with the Response. The valid range is 1 to 480. Since the total memory space of the Measurement database is 24576 bytes, the offset and read-size parameters combined must not exceed 24576.

Command layout:



5.23.2 Response

The `Response` will be delivered by the **MISO_CMD** `Characteristic`. The PoolLab2 will first emit a **MISO_SIG** `Notification` and the **response-type** will be `TYPE_READMISO` unless an error occurrs.

0	1	2	3	4	5	6	7
0x42	status-code	read-length		0x00	0x00	0x00	0x00

Bytes 2 and 3 are the 16-bit unsigned integer representation of the `read-length`, which is the amount of bytes available to be read by the central device from the PoolLab2's **MISO_CMD** `Characteristic`.

Unless an error occurs, the `read-length` will be the same as the `read-size` input-parameter, and the **MISO_CMD** `Characteristic` contains a subset of the memory `Measurement` database, at the requested range.

5.23.3 Measurement struture

`Measurements` are 24 bytes long data structures. They are made up of a `Source ID`, a `Status`, the `Parameter ID`, a `Timestamp`, and the measured `Value`.
`Source ID` is an 8-bit unsigned integer that is the index of the `Source` that was selected when this measurement was made.
`Status` is an 8-bit unsigned integer that is set to `0x01` if the measurement value exceeded the parameter's valid range, and set to `0x00` otherwise.
`Parameter ID` is 16-bit little-endian unsigned integer that indicates the measured parameter.
`Timestamp` is a 64-bit little-endian unsigned integer representing the *Epoch time* when this measurement was taken.
`Value` is a 4-byte little-endian IEEE 754 floating point number, the measurement result value.

For more information regarding the `Parameter ID`, please visit `https://poollab.org/v2/en/parameter-list`.

The memory layout of a `Measurement` is given below, the `Measurement` database is an array of 1024 of these structures.

0	1	2	3	4	5	6	7
Source ID	Status	Parameter ID		*reserved*			

8	9	10	11	12	13	14	15
Timestamp							

16	17	18	19	20	21	22	23
Value				*reserved*			

A simple way to read all available `Measurements` would be to first determine the amount of `Measurements` saved on the device by using the `GET_QUICK_INFO` command (the `NUM_MEAS` response-field), and then to execute as many `GET_MEASUREMENTS` commands with a constant `read-size` of 24, and the `offset` parameter starting at 0 and increasing by 24 with each subsequent `GET_MEASUREMENT` command.

Since the `GET_MEASUREMENTS` command supports `read-size`s up to 480 byte, up to 20 `Measurements` can be read with a single `GET_MEASUREMENTS` command. This is heavily recommended to maximize transfer-speed efficiency.

Care should be taken not to read (or at least, not to parse) data returned by this command past the end of data stored in the database.

5.24 CLEAR_MEASUREMENTS Command

Deletes all measurement data from the PoolLab2.

Command: CLEAR_MEASUREMENTS
Hex Value: 0x22
Parameters: -
Minimum FW Version: 1

5.24.1 Input

Example input into the **MOSI_CMD** Characteristic.

0	1	2	3	4	5	6	7
0x22	-	-	-	-	-	-	-

5.24.2 Response

The Response will be delivered by the **MISO_SIG** Notification and the **response-type** will always be TYPE_SIMPLE.

0	1	2	3	4	5	6	7
0x40	status-code	-	-	-	-	-	-

6 Document Revision

Revision	Date	Author	Changes
1	**17.10.2023**	**M.Opheiden**	*Pre-release without OTA commands*
2	**03.01.2024**	**M.Opheiden**	*Added note regarding Parameter ID request*
3	**26.01.2024**	**M.Opheiden**	*Changed URL for Parameter ID request*

Table 6: Document Revision History

7 Appendix

References

[1] Texas Instruments. Generic access profile (gap), 2016. Accessed: 2023-10-16. URL: https://software-dl.ti.com/lprf/simplelink_cc2640r2_latest/docs/blestack/ble_user_guide/html/ble-stack-3.x/gap.html#:~:text=The%20GAP%20layer%20of%20the,See%20GAP%20State%20Diagram.