TEP110/120 Communication Protocol Specification

BT Interface specification

Version	Author	Comment	Date
1.0	Larcher Nils (BCDS/ENG2)	Initial Version	12 Jan 2017
1.1	Larcher Nils (BCDS/ENG2)	Changes in chapter numbering and minor corrections Adpated Disclaimer	24 Jan 2017
1.2	Larcher Nils (BCDS/ENG2)	Explanations for autocalibration message exchange	25 Jan 2017
1.3	Larcher Nils (BCDS/ENG2)	Chapter on error codes introduced, Error message for PHYD added	30 Jan 2017

Note Regarding naming:

This document refers to the TEP devices and their software as TEP120. However at the moment of writing it is also valid for the devices referred to as TEP110.

Content

1. Bluetooth specification

This documentation will cover all Bluetooth connection protocol for TEP110 (eCall only) and TEP120 (eCall + PHYD) feature.

TEP120 communicates to a smartphone via Bluetooth. To do this iOS Bluetooth Low Energy (BLE) and Android Bluetooth BR / EDR (hereinafter referred to as BT Classic) are used in firmware version 10.2 (and below).

The following subchapters describe specific points of each of the two Bluetooth protocols. All subsequent chapters are not dependent on the Bluetooth protocol and therefore valid for both platforms. If there are differences in individual cases this is to be explicitly stated.

An overview of the Bluetooth messages can be found in chapter 6. Bluetooth message catalogue

1.1 Android - BT Classic

Bluetooth BR/EDR (Classic) 2.1. or higher is used with Android.

To enable this communication between two devices via Bluetooth, Bluetooth must be enabled on the smartphone and an initiator-acceptor interaction is required. A smartphone app represents the initiator side, the TEP the acceptor side.

A smartphone should continuously poll paired TEP devices to find one in range to build a valid socket. It is expected that the socket is established on the basis of RFComm to UUID.

Once a socket is successfully established to a TEP via Bluetooth, interaction with the connector is created based on input and output streams (buffered streams).

1.1.1 UUID

The TEP120 holds the following UUID

00001101-0000-1000-8000-00805F9B34FB

1.1.2 Device name

UMS Android <last 3 digits of the Bluetooth MAC hexadecimal> e.g. “UMS Android 2FB”

1.1.3 Class of Device/Service (CoD)

https://www.bluetooth.org/en-us/specification/assigned-numbers/baseband

http://bluetooth-pentest.narod.ru/software/bluetooth_class_of_device-service_generator.html

CoD 0x100 = Computer

1.1.4 Pairing

Pairing (coupling) of the smartphone and TEP is entirely at Android level.

The pairing request always originates from the smartphone. After “power-on”, the connector enables visibility mode for pairing (discoverable/pairable) for 10 minutes. After this time, a connector that is not coupled cannot be found by smartphone Bluetooth search and pairing is no longer possible. Already paired devices can still be connected after this time. This feature is only supported for Bluetooth Classic, unknown BLE devices can be linked even after the 10 minutes have elapsed.

If the connector is already connected to another smartphone pairing is not possible.
At initial pairing the TEP saves the MAC address of the smartphone. The first user that links and registers with an empty TEP is always the main user. If there is no registration after pairing, this pairing is overwritten with the next pairing.

Should another pairing and registering take place with another smartphone, this is saved as a guest user with the corresponding MAC address.

1.2 iOS - BT LE

Bluetooth LE (BLE) is used with iOS.

The “higher” protocol is identical to the Android version, except the principle communication is not socket-based, but runs on the reading and writing of the characteristics of a service.

1.2.1 Device name

UMS iPhone <last 3 digits of the Bluetooth MAC of the BT Classic Device, hexadecimal>, e.g. “UMS iPhone 2FB”

1.2.2 General

iOS allows socket access to BLE devices only within the Mfi program (Made for iPhone).

Nevertheless, in order to maintain the protocol, the following options are available:

- TEP120 offers one or more services
- Within the service there is a characteristic that is used for reading and writing from the iPhone and TEP120

1.2.3 UUIDs

The following UUIDs have been specified to enable the devices to communicate with each other:

Name	UUID
Service (contains all characteristics)	58d79b40-bc2e-11e4-b205-0002a5d5c51b
Read Characteristics (Property: Indicate)	645706e0-bc2e-11e4-97e7-0002a5d5c51b
Write Characteristics (Property: Write)	6a6e79a0-bc2e-11e4-bad4-0002a5d5c51b

1.2.4 Pairing

A pairing as with Bluetooth Classic is not given with BLE, in principle, each central can be linked with each peripheral as long as the UUID is known.

The pairing is not established on the system, but should be implemented within an app according to the following scheme:

- Scan for TEP120 devices
- Display of devices found
- User selects a device, the app then provides the connection and registers the connector

The connection between the smartphone and TEP120 is unencrypted, so a pairing dialog on the smartphone is omitted. A reconnect after the connection is ended is possible at any time without any intervention by the user.

If the iOS app is removed or terminated in the background, it must send a “Disconnect” to the TEP120, so that the TEP120 can switch to the correct flashing mode.

2 Connection Handling

The following chapter is needed to communicate via Bluetooth (Classic + BLE) with the TEP120

In general it is assumed that an app running on a smartphone connects to the TEP120 but in general any device which is capable to communicate via Bluetooth can be used to connect to the TEP120.

2.1 Message types

For communication purposes the selected communication protocol is MQTT.

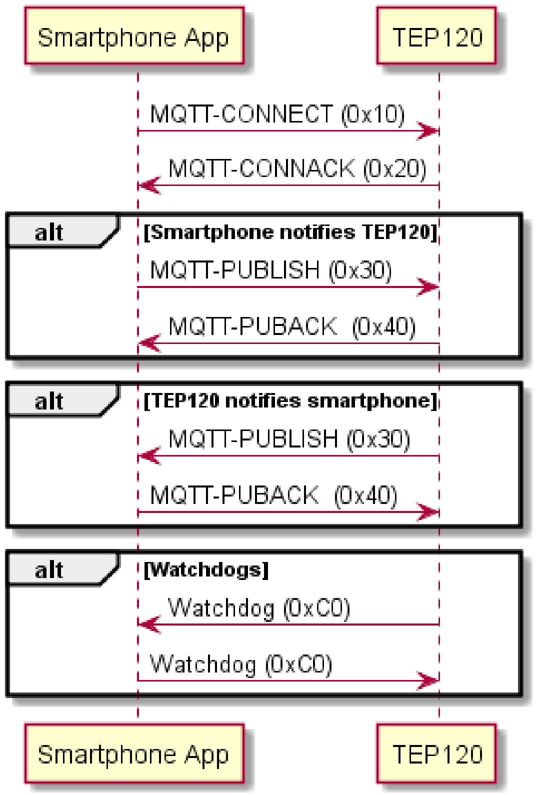
There are 4 different MQTT messages defined:

- MQTT-CONNECT
- MQTT-CONNACK
- MQTT-PUBLISH
- MQTT-PUBACK

Additionally watchdog messages have to be exchanged between TEP120 and Smartphone app and vice versa. The watchdog is used to check if the connection between the TEP120 and the smartphone is still available and working.

All information exchanged between the smartphone app and the TEP120 are transported on either the MQTT or the watchdog messages.

The sequence diagram looks like the following,.



A complete overview of all exchanged messages can be found in chapter 6.

2.2 Connection/Operating Modes

The TEP120 can be set up to run in one of the modes defined below:

#	ModeName	Description
1	Ecall	The TEP120 will send a crash message if it has detected a crash

2	Demo Ecall	Same as mode 1 but with reduced thresholds to trigger a crash event easily
3	Pay How You Drive (PHYD)	In this mode the driving behavior is evaluated and driving event messages are triggered accordingly
4	Combined	Mode 1 + 3 running at the same time
9	Setup	Mode for user management and to switch in between modes

2.3 Connection Scenarios

Should the Account ID or MAC address (with BLE devices equivalent to the Phone ID) already be stored in the TEP120 user list then the TEP120 compares the data given in the connect message with the stored data.

Depending on the scenario the following return codes (byte <RC>) are possible:

Scenario	Connect Message	Stored in TEP120	Action to perform on the TEP120	Return Code <RC>	
1. User has reinstalled the app and re-registers the TEP120.	Main/guest user	identical	The TEP120 overwrites the entry with the new data	0	<0x00>
	AccountID	identical			
	MAC-Adresse/Phone ID	identical			
	TgicCompanyID	identical			
	RegDate	older			
2. User has changed the smartphone and re-registers the TEP120.	Main/guest user	identical	The TEP120 overwrites the entry with the new data.	1	<0x01>
	AccountID	identical			
	MAC-Adresse/Phone ID	different			
	TgicCompanyID	identical			
	RegDate	older			
3. User registers with its existing app with another AccountID.	Main/guest user	identical	The TEP120 overwrites the entry with the new data	2	<0x02>
	AccountID	different			
	MAC-Adresse/Phone ID	identical			
	TgicCompanyID	identical			
	RegDate	older			
4. Users registers on the same smartphone with a new app (other insurance) with a new account.	Main/guest user	identical	The TEP120 overwrites the entry with the new data.	3	<0x03>
	AccountID	different			
	MAC-Adresse/Phone ID	identical			
	TgicCompanyID	different			
	RegDate	older			
5. A connect attempt by a known iPhone is rejected	Main/guest user	identical	The TEP120 rejects the connection	4	<0x04>
	AccountID	identical			

	MAC-Adresse/Phone ID	identical			
	TgicCompanyID	identical			
	RegDate	identical			
6. Normal connection of an app with an enabled account	Main/guest user	identical	No action	0	<0x00>
	AccountID	identical			
	MAC-Adresse/Phone ID	identical			
	TgicCompanyID	identical			
	RegDate	identical			
7. User has successfully registered an app of another insurance on the same smartphone. Now the first connection with the old app takes place, which has not been disabled by the insurance company. This scenario can also occur with guest users, if the guest user account did not receive an erase-PUSH	Main/guest user	identical	The TEP120 rejects the connection	-2	<0xFE>
	AccountID	different			
	MAC-Adresse/Phone ID	identical			
	TgicCompanyID	different			
	RegDate	newer			
8. Connector is full	x	x	The TEP120 rejects the connection	-3	<0xFD>
9. First ever user who wants to register connector is a guest user and the sensor still does not have a main user.	x	x	The TEP120 rejects the connection	-4	<0xFC>
10. Main user memory space is occupied and a second main user tries to connect.	Main/guest user	Main user	The TEP120 rejects the connection	-5	<0xFB>
	AccountID	different			
	MAC-Adresse/Phone ID	different			
	TgicCompanyID	irrelevant			
	RegDate	irrelevant			
11. TEP120 functional test unsuccessful	x	x	See Chapter 5		
	All other combinations		The TEP120 rejects the connection	-1	<0xFF>

3 User management

The number of registered users on an TEP120 is limited to 5 (4 guest users + 1 main user). However, this does not apply to the number of user accounts and UMD connectors registered in the app.

Obviously, the possible time from the TEP120 “power-on” or coming within range until the connection has been established becomes longer with the increasing number of connectors registered with the app. Although the number of registered connectors is not limited, as a rule it is assumed that a realistic number is < 10. For 10 active accounts with registered connectors, time until connection is established is approximately 13 seconds. The PUBLISH messages in the following chapters are always answered with a PUBACK. A positive PUBACK is only possible if the app on the smartphone has previously connected to the TEP120 in the setup mode (MQTT-Connect Mode 9).

3.1 Main user-before-guest user problem

- Scenario: With several participants in a vehicle all trying to connect with the same connector - can the connector identify the “correct” one from several connection attempts and reject all the others?

The solution to this problem is implemented thus: The guest user is rejected at the first connect attempt (from the perspective of the app), but is added to a waiting list in the connector.

If, within one power cycle[1] of the TEP120 a new connection attempt is made by the same user **without** a main user joining in the meantime, the user is admitted.

If a main user has connected in the meantime, the guest user is denied.

The guest user must maintain a minimum waiting time of **5 seconds** between connection attempts.

The first rejection of a guest user differs between Android and iOS devices:

- The Android smartphone is rejected directly on attempting a Bluetooth connection, so that Bluetooth connectivity is not established and a MQTT-Connect cannot be sent

- With iOS devices, the Bluetooth connection is allowed and only after the MQTT-Connect is sent and the user is identified as a guest user on TEP120 is a MQTT-Connack with return code 4 sent to the smartphone and then a Bluetooth disconnect is performed.

4 Triggering crash events

4.1 Automatic trigger

If the TEP120 detects [3] an accident (crash status changes from 0 to 1), it sends crash messages to the smartphone .
An accident progress consists of several consecutive messages with status 1-4 and is terminated by a message with the status 0, then no further crash messages are sent. The smartphone app should note the message with the highest status (the highest severity index) and also the time of the triggered event to avoid multiple triggers. the recommendation is to wait 60 seconds until a new crash event can be triggered. The TEP120 will also only deliver consecutive values in ascending order (1,1,1,2,2,3,3,0).

If no transaction is made with a status 0 message, the connection was probably disrupted, and the app takes on status 4 (highest severity index 100). The smartphone app can consider an accident history to be completed after 3 seconds.

Whilst the crash is happening the TEP120 does not send any watchdog message (see Chapter 5.2). The crash messages are broadcast messages, therefore acknowledgement from the smartphone app is not expected. The TEP120 continuously calculates the CI (crash severity index) and the crash status (CS), then transmits every fourth message and the final 0 message. This results in the following example sequence of the message counter <CNT>: 0, 4, 8, 12, 13.

If the connection is interrupted during a crash the TEP120 saves the final crash message. If the connection can be restored within 15 seconds, the TEP120 sends this message and the final 0 message without expecting a MQTT-Connect.

4.2 Manual Trigger

A crash event could also be triggered manually. This option can be implemented in the smartphone app.

5 LED Behavior

The LED can give some basic information to the user about the current state of the TEP120.

There are 2 blink codes which can be used by the app developer/provider for his own purpose.

Blink code (continuous)	Current status
Off	No power supply
Off	The TEP120 is in power-save mode
3 s off – 1 s on	TEP120 is ready to connect; No smartphone currently connected
0.5 s off - 0.5 s on	The TEP120 has an error (see Error codes chapter)
3 s off - 3 s on	The TEP120 is in setup mode or the smartphone app has set blink code 0x01
1.5 s off - 1.5 s on	The smartphone app has set blink code 0x02
On	TEP120 connected, ready and no error set or recognized

5.1 Blink priorities

1.) Internal error in the connector (half a second on, half a second off)

2.) No smartphone connected (3 seconds on, 1 second off)

3.) Flashing by the app (see above) or TEP120 in setup mode (3 seconds on, 3 seconds off)

4.) TEP120 ready and no error set or recognized (steady)

5.2 Blink codes from smartphone app

When the TEP120 is in mode 1 or 2, the smartphone app can send its error to the connector and affect the normal blinking behavior.
When the TEP120 is in the setup mode, transmission of a blink code with return code 0xF1 (wrong mode) is acknowledged and the blinking behavior does not change.

Message structure

Bytes	Explanation
<0x30>	Retain = false | Type = PUBLISH | Duplicated = false | QoS = LOW
<0x02>	Length of following message
<0x45> (‘E’)	Error
<BLINKCODE>	1 Byte, see below

Blink Codes

The following blink codes are available:

- blink code = 0x00: do not blink (blink in response to connector behavior)
- blink code = 0x01: blink sequence 1 (3 seconds on, 3 seconds off)
- blink code = 0x02: blink sequence 2 (1.5 seconds on, 1.5 seconds off)

After a disconnect, reset or power off/on again, blinking in response to connector behavior is once again enabled.

This message is also acknowledged with a PUBACK (see Chapter 3).

6. Pay How you drive features

6. Bluetooth Message catalogue

6.1 Connection Handling

6.1.1 Connection setup

If the app could successfully establish a socket connection with the TEP120 the app sends a “MQTT -Connect”.

This is structured as follows:

6.1.1.1 Android smartphones

Bytes	Explanation
<0x10>	Retain = false | Type = CONNECT | Duplicated = false | QoS = LOW
<0x16>	Length of following message
<MODE>	Mode of operation, see details below (1 Byte)
<USER_FLAG>	(1 Byte)
<ACCOUNT_ID>	User Account ID; ASCII encoded (8 Byte)
<TGIC_COMPANY_ID>	Unique ID of insurance company; int32 (4 Byte)
<REG_DATE>	Registration Date; Long 64 bit in UNIX Epoch format (8 Byte)

The following valid operation modes can be transmitted:

- 0x01 --> Ecall mode
- 0x02 --> Ecall mode with adapted threshold for Ecall triggering (Demo usecase)
- 0x03 --> PHYD mode
- 0x04 --> Combined mode (PHYD + Ecall)
- 0x09 --> Mode for user management

The following valid options exist for the <USER_FLAG>:

- 0x48 --> 'H' for main user
- 0x47--> 'G' for guest user

6.1.1.2 iOS smartphones

For iOS smartphones MQTT-CONNECT appears as follows:

Bytes	Explanation
<0x10>	Retain = false | Type = CONNECT | Duplicated = false | QoS = LOW
<0x16>	Length of following message
<MODE>	Mode of operation, see details below (1 Byte)
<USER_FLAG>	(1 Byte)
<ACCOUNT_ID>	User Account ID; ASCII encoded (8 Byte)
<TGIC_COMPANY_ID>	Unique ID of insurance company; int32 (4 Byte)
<REG_DATE>	Registration Date; Long 64 bit in UNIX Epoch format (8 Byte)
<PHONE ID>	Phone Number, ASCII encoded (16 Byte), Format: +490123456798

6.1.2 Connection Response

The TEP120 responds with an MQTT-CONNACK. This is structured as follows:

Bytes	Explanation
<0x20>	Retain=false | Type=CONNACK | Duplicated=false | QoS=LOW
<0x0C>	Length of following message
<STATUS>	Current mode or error code (1 Byte), see chapter Self Diagnosis
<RC>	Return code (1 Byte), see below
<UID>	Unique ID; MAC address of TEP120 device (6 Bytes)
<HW-VERSION>	Hardware Version (1 Byte)
<SW-VERSION_MAJOR>	Software Major Version (1 Byte)
<SW-VERSION_MINOR>	Software Version Minor (1 Byte)
<PROTOCOL-VERSION>	Communication protocol version (1 Byte)

6.2. User Management

6.2.1 Deleting guest user

Each guest user is authorized to delete themselves, but they are not authorized to delete other guest users or particularly the main user. The main user is always authorized to delete guest users. After receiving a positive PUBACK, the TEP120 deletes the Bluetooth connection to the smartphone and is then visible to other devices.

Message structure

Bytes	Explanation
<0x30>	Retain = false | Type = PUBLISH | Duplicated = false | QoS = LOW
<0x09>	Length of following message
<0x47> (‘G’)	Guest
<ACCOUNT_ID>	User Account ID; ASCII encoded (8 Byte)

In addition to the general return codes in PUBACK this PUBLISH can still be acknowledged with the following return code:

Bytes	Provided with a sign	Description
<0xEB>	-21	Account ID does not exist

6.2.2 Deleting all guest users

A main user can delete all guest users at one time. A guest user is not allowed to send this PUBLISH message. If this is the case, a negative PUBACK is the response, not even the own guest user account is deleted.

Message structure

Bytes	Explanation
<0x30>	Retain = false | Type = PUBLISH | Duplicated = false | QoS = LOW
<0x01>	Length of following message
<0x41> (‘A’)	**A**ll Guest users

6.2.3 Reset the TEP120

This command allows the main user to delete all users stored on the TEP120 - including themselves. Guest users are not authorized to perform this action. After receiving a positive PUBACK the TEP120 triggers a system reset (disconnects the existing Bluetooth connection) and is then visible to other devices.

Message structure

Bytes	Explanation
<0x30>	Retain = false | Type = PUBLISH | Duplicated = false | QoS = LOW
<0x01>	Length of following message
<0x5A> (‘Z’)	

6.2.4 TEP response for PUBLISH messages

The PUBLISH messages in the previous chapters are always answered with a PUBACK message which contains the following information:

Bytes	Explanation
<0x40>	Retain = false | Type = PUBACK | Duplicated = false | QoS = LOW
<0x01>	Length of following message
<RC>	Return code (1 Byte)

In all the following cases the return code (<RC>) may include the following values:

Bytes	Provided with a sign	Description
<0x00>	0	Normal / positive
<0xF1>	-15	Setup action in normal mode initiated
<0xF0>	-16	Not authorized
<0xED>	-19	Unknown error

6.3. Software Update

All users are authorized to perform a software update in setup mode. The update is encrypted and signed.

The TEP120 has 2 partitions, the old software is automatically started should an update process be interrupted or fail. Then the existing Bluetooth connection is disconnected.

Message structure

Bytes	Explanation
<0x30>	Retain = false | Type = PUBLISH | Duplicated = false | QoS = LOW
<0x01>	Length of following message
<0x46> (‘F’)	Flash Request

In addition to the general return codes in PUBACK this PUBLISH can still be acknowledged with the following return code:

Bytes	Provided with a sign	Description
<0x60>	96	Update binary must be transmitted for Partition 0
<0x61>	97	Update binary must be transmitted for Partition 1

After that, transmission of the expected update binary must be started within the next 2 minutes. The update is transmitted in packets; one packet consists of 2 bytes packet counters and 128 bytes of the update binary. The TEP120 acknowledges each packet with the transmitted packet counter if processing was successful.

No more than 30 seconds must elapse between the transmissions of these packets.

The TEP120 sends a final PUBLISH message, which is structured as follows:

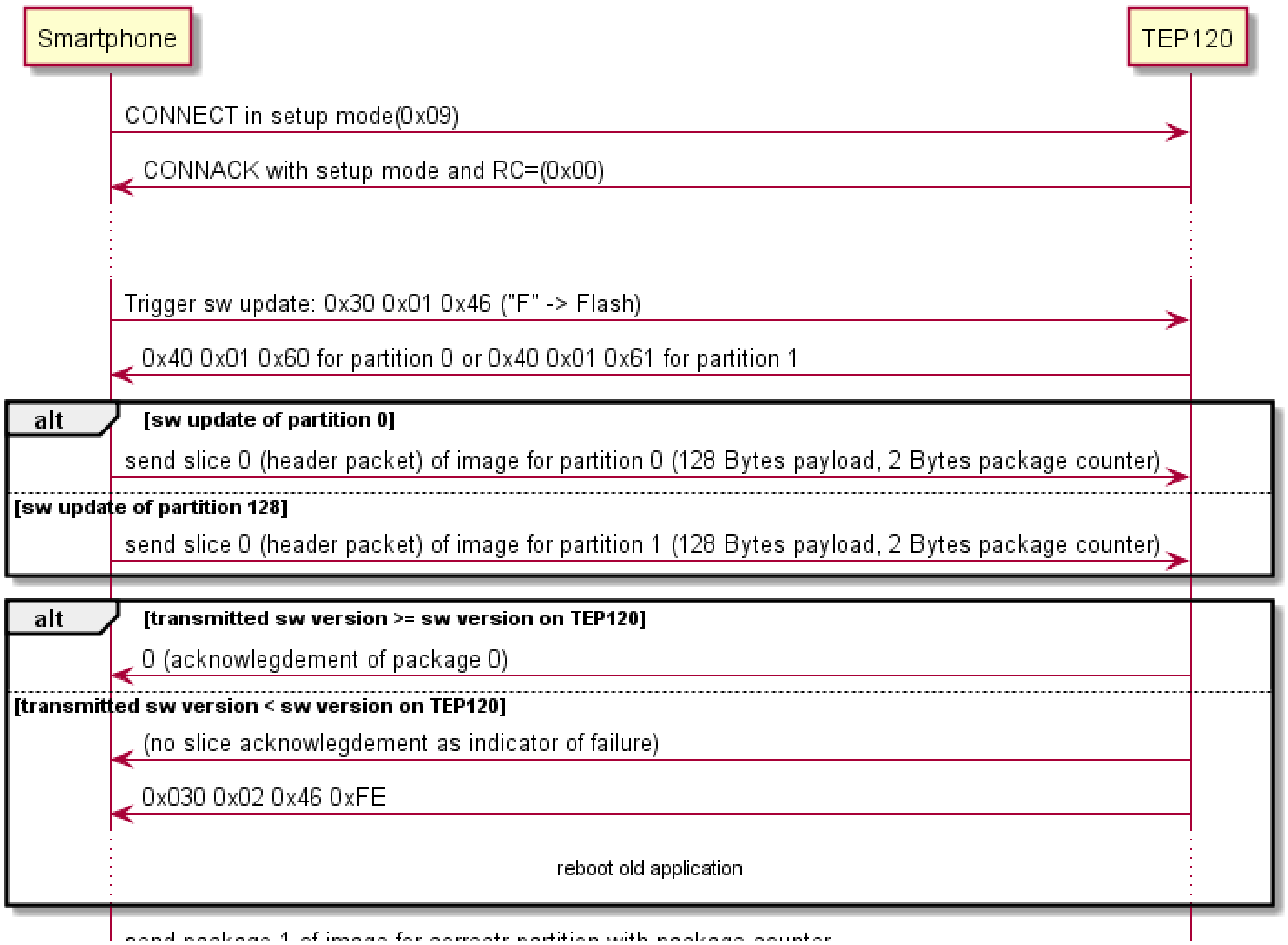
Bytes	Explanation
<0x30>	Retain = false | Type = PUBLISH | Duplicated = false | QoS = LOW
<0x02>	Length of following message
<0x46> (‘F’)	Flash
<RC>	Return code (1 Byte)

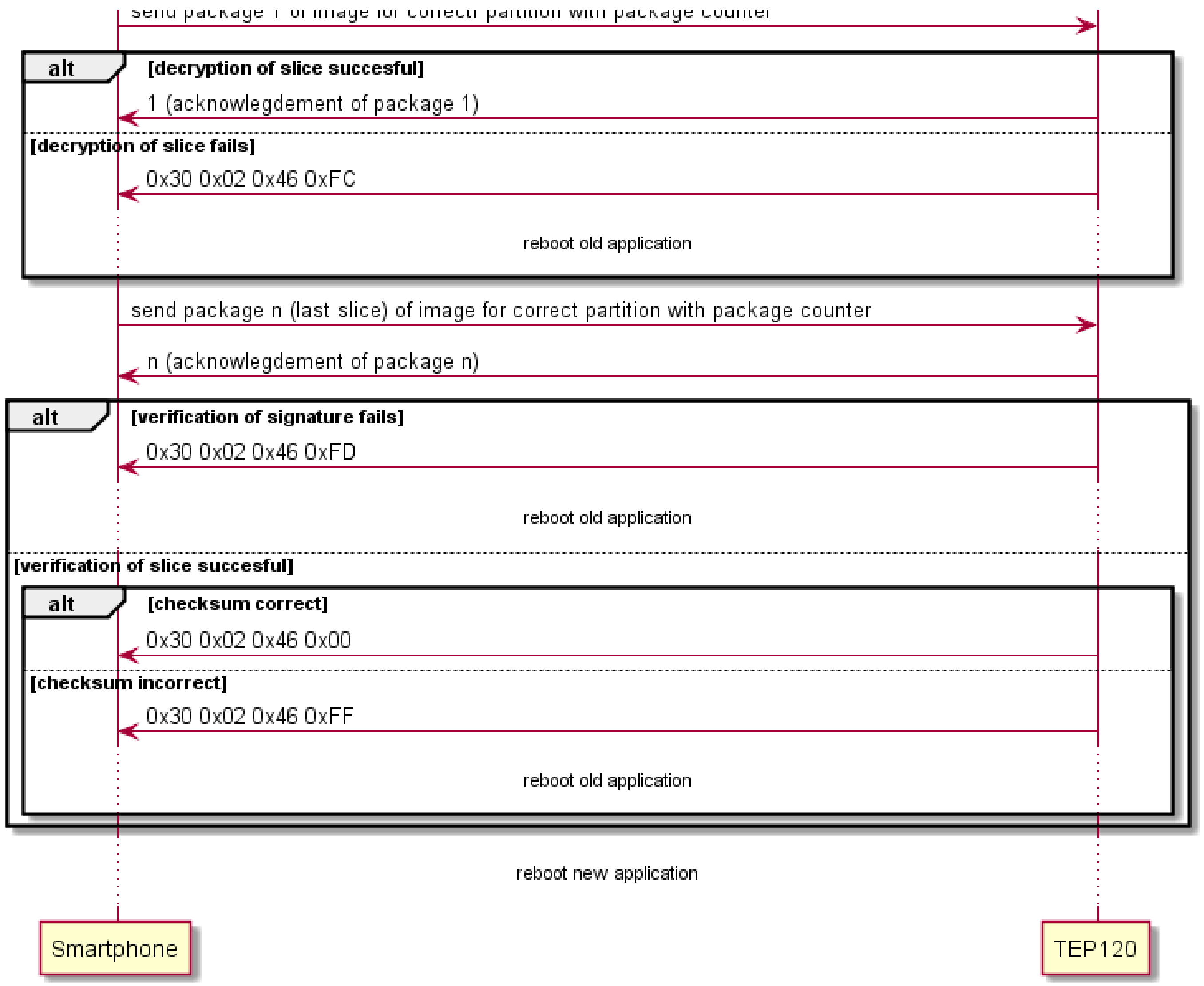
The following return codes can occur in this context:

Bytes	Provided with a sign	Description
<0x00>	0	Software update was successful
<0xFF>	-1	Checksum of the binary invalid
<0xFE>	-2	Software version older than that currently installed
<0xFD>	-3	Verification of the update failed
<0xFC>	-4	Decryption of a packet failed

Depending on the success of the update process, then either the old or new TEP120 software is started.

The following diagram shows the update sequence in detail.


Smartphone
TEP120
CONNECT in setup mode(0x09)
CONNACK with setup mode and RC=(0x00)
Trigger sw update: 0x30 0x01 0x46 ("F" -> Flash)
0x40 0x01 0x60 for partition 0 or 0x40 0x01 0x61 for partition 1
alt
[sw update of partition 0]
send slice 0 (header packet) of image for partition 0 (128 Bytes payload, 2 Bytes package counter)
[sw update of partition 128]
send slice 0 (header packet) of image for partition 1 (128 Bytes payload, 2 Bytes package counter)
alt
[transmitted sw version >= sw version on TEP120]
0 (acknowlegdement of package 0)
[transmitted sw version < sw version on TEP120]
(no slice acknowlegdement as indicator of failure)
0x030 0x02 0x46 0xFE
reboot old application
send package 1 of image for correctr partition with package counter


send package 1 of image for correct partition with package counter
alt
[decryption of slice succesful]
1 (acknowlegdement of package 1)
[decryption of slice fails]
0x30 0x02 0x46 0xFC
reboot old application
send package n (last slice) of image for correct partition with package counter
n (acknowlegdement of package n)
alt
[verification of signature fails]
0x30 0x02 0x46 0xFD
reboot old application
[verification of slice succesful]
alt
[checksum correct]
0x30 0x02 0x46 0x00
[checksum incorrect]
0x30 0x02 0x46 0xFF
reboot old application
reboot new application
Smartphone
TEP120

6.4. Error management

Information about possible errors are accessible to the user, either by the TEP120 LED, in a MQTT-CONNACK message, or via the watchdog message.

6.4.1 Watchdog from the smartphone

To ensure a stable Bluetooth connection, the smartphone periodically sends a watchdog to the TEP120. A valid message (see below) is received and checked by the TEP120. The TEP120 performs a restart if the TEP120 is connected to another device and this watchdog is absent for 30 seconds.

Message structure

Bytes	Explanation
<0xC0>	Retain = false | Type = PING | Duplicated = false | QoS = LOW
<0x00>	Length of following message

6.4.2 Watchdog from the TEP120

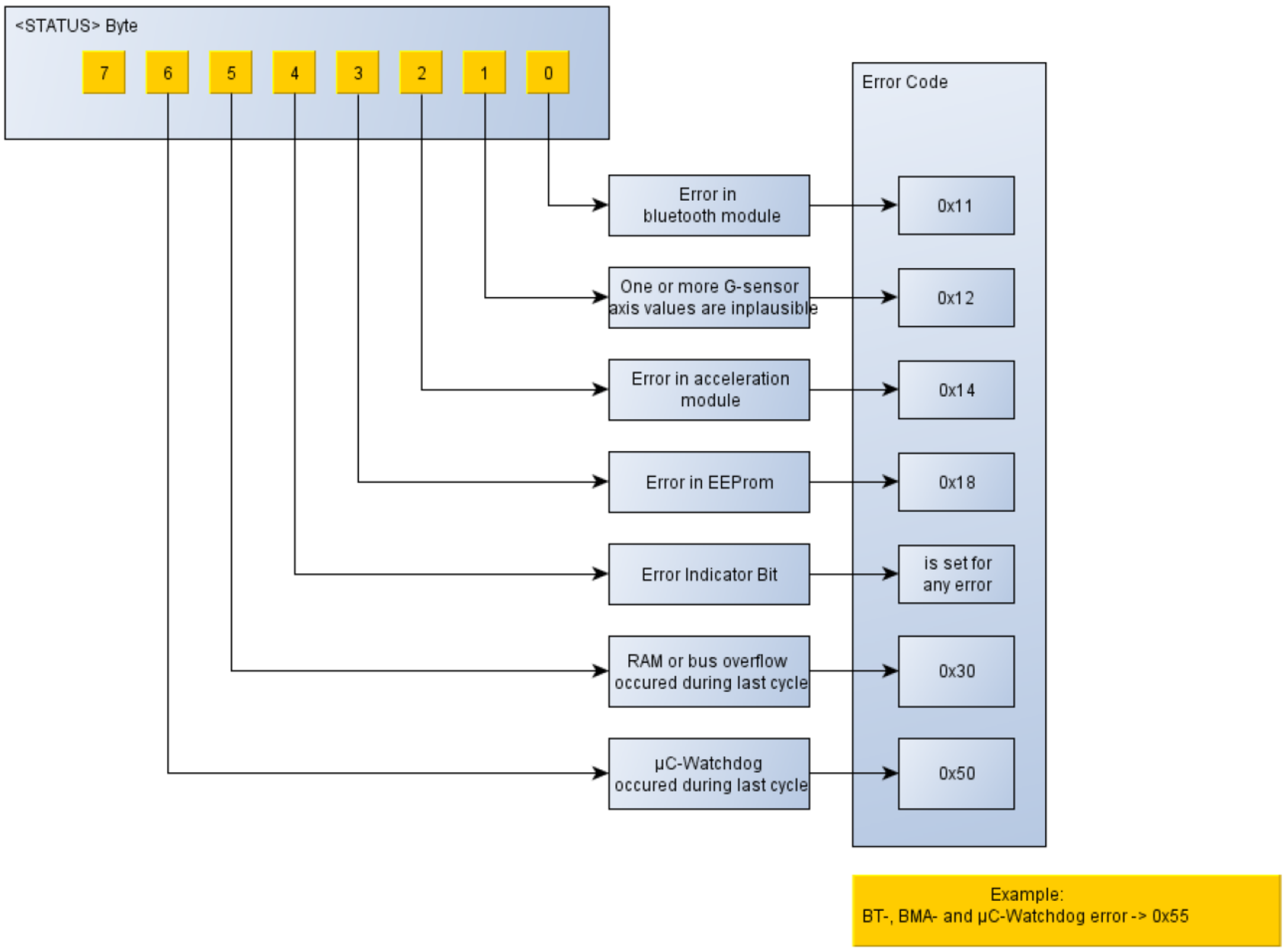
After a connection between the TEP120 and a Bluetooth device has been established, the TEP120 continuously sends messages to the other party every 2 seconds. These messages include the internal status of the TEP120 and the implicit information that the TEP120 can still send messages. Errors and problems within the TEP120 are transmitted through the status messages to the other party.

Message structure

Bytes	Explanation
<0xC0>	Retain = false | Type = PING | Duplicated = false | QoS = LOW
<0x01>	Length of following message
<STATUS>	Current mode or error code (see figure below)

6.4.2.1 Error codes in watchdog and their context

The watchdog status byte can be decoded as follows:


<STATUS> Byte
7
6
5
4
3
2
1
0
Error Code
Error in bluetooth module
0x11
One or more G-sensor axis values are inplausible
0x12
Error in acceleration module
0x14
Error in EEProm
0x18
Error Indicator Bit
is set for any error
RAM or bus overflow occured during last cycle
0x30
µC-Watchdog occured during last cycle
0x50
Example:
BT-, BMA- and µC-Watchdog error -> 0x55

There are the following error situations:

Error Context	Error Description
Bluetooth chip	The Bluetooth chip in the TEP 120 is no longer working properly.
3-axis sensor	The 3-axis sensor in the TEP120 delivers implausible values.

Accelerometer	The accelerometer in TEP120 is no longer working properly.
EEProm (memory)	An error has occurred with the internal memory of the TEP120.
Memory overflow	A memory overflow has occurred in the TEP120.
Error context	Error description
µC watchdog	A system error has caused a restart of theTEP120. Please disconnect the connector from the power supply and then reconnect. If the error occurs again, the connector must be sent for repair.
Send the sensor for repair	Please send the TEP120 in for repair.
Unknown	Receiving an unknown error code from the TEP120.
Absent watchdog message	The TEP120 sends no system monitoring messages. Please check the condition of the connector.

6.4.3 Error Codes

An overview of error codes whioch can be sent from the TEP120 to the smartphone app.

To get a full understanding of the cause for the errors the according chapters have to be checked.

Connection Setup:

Bytes	with sign	Description
<0xFF>	-1	TEP120 functional test unsuccessful
<0xFE>	-2	Other insurance already registered
<0xFD>	-3	Connector is full
<0xFC>	-4	First ever user who wants to register connector is a guest user and the sensor still does not have a main user
<0xFB>	-5	Main user memory space is occupied and a second main user tries to connect.

Firmware Update:

Bytes	with sign	Description
<0xFF>	-1	Checksum of the binary invalid
<0xFE>	-2	Software version older than that currently installed
<0xFD>	-3	Verification of the update failed
<0xFC>	-4	Decryption of a packet failed

PHYD:

Bytes	with sign	Description
<0xF2>	-14	PHYD action triggered outside of PHYD/combined mode

6.5. Messages for crash detection

Message structure

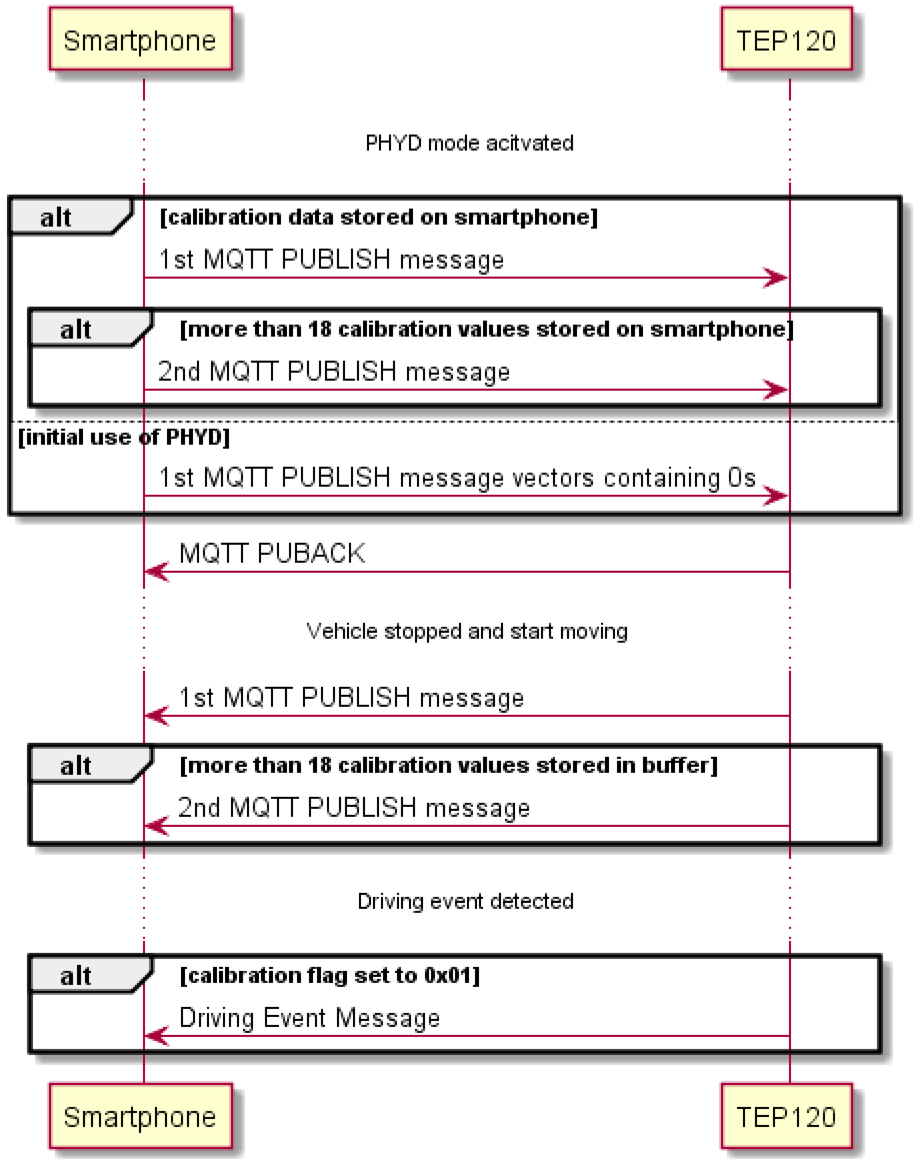
Bytes	Explanation
<0x30>	Retain = false | Type = PUBLISH | Duplicated = false | QoS = LOW
<0x05>	Length of following message
<0x43> ('C')	Crash
<EVENT_ID>	ID of crash event
<CNT>	Number of message in crash event
<CI>	Crash Index (0-100)
<CS>	Crash Status (0-4)

Crash severity Index:

CS	CI[2]	Description
0	TBD.	no crash
1	2	
2	> 2	normal vehicle behavior left (might indicate begin of crash)
3	> 40	medium crash
4	> 50	massive crash

6.6. Messages for Pay How You Drive with Autocalibration

Message Exchange sequence:


Smartphone
TEP120
PHYD mode acitvated
alt
[calibration data stored on smartphone]
1st MQTT PUBLISH message
alt
[more than 18 calibration values stored on smartphone]
2nd MQTT PUBLISH message
[initial use of PHYD]
1st MQTT PUBLISH message vectors containing 0s
MQTT PUBACK
Vehicle stopped and start moving
1st MQTT PUBLISH message
alt
[more than 18 calibration values stored in buffer]
2nd MQTT PUBLISH message
Driving event detected
alt
[calibration flag set to 0x01]
Driving Event Message
Smartphone
TEP120

6.6.1 Messages to load autocalibration values

The calibration message is used to transfer calibration data between the TEP120 and the smartphone. The smartphone is used as a storage to avoid to wait for calibration the next time the PHYD is activated.Therefore the smartphone will send stored calibration data to the TEP120 at activation of PHYD and the TEP120 will send an update with stored values to the smartphone everytime the calculations for calibration were executed.

If the buffer is populated with more than 18 vectors one single MQTT PUBLISH message would be too small. In this case a second PUBLISH message, containing the rest of the buffer is sent from the smartphone to the TEP120

Structure of the 1st MQTT PUBLISH message:

Bytes	Explanation
<0x30>	Retain=false | Type=PUBLISH | Duplicated=false | QoS=LOW
<LEN>	Length of payload 1 - 255 Bytes (set to 255 if 2nd message has to expected)
<’V’>	Message to transmit Calibration Values
<V_GRAVITY>	Gravity Vector (3 x FLOAT, IEEE754, 12 Byte)

<V_LONGITUDE>	Longitudinal Vector (3 x FLOAT, IEEE754, 12 Byte)
<CALIBRATION_FLAG>	Calibrations Status (either 0x00 for not calibrated or 0x01)
<B_SIZE>	Number of Rows in Matrix/ Vectors in Buffer (up to 20 vectors)
<INDEX>	Buffer Index
<VALUES>	Vectors in Buffer (each 3 x FLOAT, IEEE754, 12 Byte)

The buffer is cut just at the very 255th byte - so not only complete vectors are transferred. In the case stated here only having both messages leads to valid data.

Structure of the 2nd MQTT PUBLISH message:

Bytes	Explanation
<0x30>	Retain=false | Type=PUBLISH | Duplicated=false | QoS=LOW
<LEN>	Length of payload 1 - 14 Bytes (max length of the second message)
<'V'>	Message to transmit Calibration Values
<VALUES>	Vectors in Buffer (each 3 x FLOAT, IEEE754, 12 Byte)

Sending the calibration values from smartphone to TEP120 is confirmed with a PUBACK once both messages have been received. The other direction has no Acknowledgement.

6.6.2 Driving Event Messages

As seen in the application overview figure there are two new modes - PHYD Mode 0x03 and Combined Mode 0x04. If TEP120 is calibrated properly and ABC-driving events are detected, the following PHYD Event message is transmitted to the connected the Smartphone:

Byte	Explanation
<0x30>	Retain=false | Type=PUBLISH | Duplicated=false | QoS=LOW
<0x03>	length of following message
<0x50> ('P')	Pay how you drive event message
<EVENT_TYPE>	Type of PHYD event
<EVENT_HEAVINESS>	Heaviness of PHYD event (one byte, unsigned integer 0-255)

Encoding of Event Type:

Event Type	Byte
No Event	0x00
Heavy Braking	0x01
Heavy Speed Up	0x02
Heavy Lateral Speed Up Right	0x03
Heavy Lateral Speed Up Left	0x04

6.6.3 Autocalibration in wrong mode error message

If a smartphone is connected in any other mode than PHYD or combined mode and last stored calibration values are sent via PUBLISH message to TEP120, the following PUBACK message will be received:

Bytes	Description

<0x40>	Retain=false | Type=PUBACK | Duplicated=false | QoS=LOW
<0x01>	length of following message
<0xF2>	-14, PHYD action triggered outside of PHYD/combined mode

[1]With some Bluetooth stacks a synchronization error exists in the stack. With a connection that is too fast this causes the Bluetooth stack to freeze internally and it is only possible to correct the error by restarting Bluetooth or the smartphone.
The ten seconds is a value based on experience from the community to get around this issue.

[2] No final values defined yet

[3]: Bluetooth LE: separate characteristic only for crash events!

So the package can still be sent in the event of power failure.