TELTONIKA | Telematics

Tel. +370 5 212 7472
Fax. +370 5 276 1380

www.teltonika-gps.com
info@teltonika.lt

BTXXXX EN12830 FUNCTIONALITY DESCRIPTION FOR CLIENT

Date: 2024.10.10

Updated: 2025.06.06

Revision: Rev.01

Contents

Tel. +370 5 212 7472
Fax. +370 5 276 1380
www.teltonika-gps.com
info@teltonika.lt

Functionality description

Features:

- Start Recording with periods of 1-, 5- and 15-minutes and time sync.
- Stop Recording.
- Delete Records.
- Download records via characteristic with 2 methods.
- Download records from a specific timestamp.
- Time synchronization.
- Encrypted commands with Challenge-Response mechanism.
- Encrypted Recording Status Info and Records Download Mechanism.

General information

All EN12830 related communication with the device is encrypted to prevent unauthorized modification. To prevent replay attacks, all commands require a challenge response mechanism.

The unencrypted random value should be read from corresponding characteristic. It should be appended to the beginning of the command. Then data that contains random value and command value should be encrypted. This encrypted data should be written to corresponding command characteristic. The result of this command can be read from same characteristic. If the command is about reading from device, the corresponding encrypted data can be read from Record Data characteristic.

Temperatures are stored in pages. Each page contains a header and can store up to 945 records. If recording into this page is started, it contains starting time. If recording is finished to this page, this header also contains stop time, record count of that page and CRC.

Functionality

Parameters

There are 2 parameters need to be given with start command. First one is timestamp, second on is period.

Timestamp is stored for user. It is not needed for temperature recording and operation of this firmware. This way, user can use any time format up to 8 bytes.

Table 1:Parameters values and types.

Characteristic UUIDs

Created new service with 4 characteristics.

Table 2:Char UUID table.

Name	Service UUID	Characteristic UUID	Read-Write	Encrypted	Size Flexible	Size
Record Info	e61c0000-7df8-4d4e-8e6d-c611745b92e9	e61c0001-7df8-4d4e-8e6d-c611745b92e9	Read Only	Yes	Fixed	16 Bytes
Random Value	e61c0000-7df8-4d4e-8e6d-c611745b92e9	e61c0002-7df8-4d4e-8e6d-c611745b92e9	Read Only	No	Fixed	2 Bytes
Record Data	e61c0000-7df8-4d4e-8e6d-c611745b92e9	e61c0003-7df8-4d4e-8e6d-c611745b92e9	Read Only	Yes	Fixed	42 Bytes
Command	e61c0000-7df8-4d4e-8e6d-c611745b92e9	e61c0004-7df8-4d4e-8e6d-c611745b92e9	Write then Read	Write: Yes Read: No	Write: Variable Read: Fixed	Write: 4 Bytes Read: 1 Byte
SHT4x Serial Number	0x180A	a610249f-913e-46bd-b14f-c6dedc432165	Read Only	No	Yes (Max 12 bytes)	String

Record Info

This characteristic contains the recent information about current recording. It holds the variables in little-endian format and it is encrypted.

- '*is_recording*' can be either 0, or 1. Recording is happening, value is 1, otherwise 0. If any reset occurs while recording, current session will stop and this value will be changed to 0.
- 'interval' holds the period of recording in unit of seconds. If '*is_recording*' is 1, this value contains the period of ongoing recording. Otherwise, it holds stored recording.
- '*number_of_records*' holds the count of records. If '*is_recording*' is 1, this value contains the number of records for recording in session. Records are written to flash after every 3 measurements. So, if a reset occurs, latest 1 or 2 record can be lost. If '*is_recording*' is 0, it holds count of records stored in flash.
- '*start_timestamp*' holds the timestamp. If '*is_recording*' is 1, this value contains the timestamp of ongoing recording. Otherwise, it holds timestamp of records stored in the flash.

```
struct {
    uint16_t is_recording;
    uint16_t interval;
    uint32_t number_of_records;
    uint32_t start_timestamp;
}
```

Figure 1. Structure of "record info" characteristic.

Table 3:Example of decrypted "record info ".

	is_recording	interval	number_of_records	start_timestamp
Received (Hex)	01-00	2C-01	3D-02	01-FD-B7-62-00-00-00-00
Meaning	1	300	573	1656225025

Random Value

This characteristic contains the little-endian uint16_t type random value that is need to be used while sending command. This value is not encrypted. Firmware compares this value with the value from decrypted command request. If it matches, command is accepted.

Record Data

This characteristic is for sending encrypted records and other data.

```
typedef struct {
    uint32_t timestamp;
    uint16_t interval;
    uint16_t CRC;
} start_config_t;
```

```
typedef struct {
    uint16_t stored_record_count;
    uint16_t CRC;
} stop_config_t;
```

```
struct {
    uint16_t index;
    start_config_t start_config;
    stop_config_t  stop_config;
    uint8_t reserved[24];
};
```

Figure 2. Structure of decrypted "record data" when index is 0 (zero).

When "START_RECORD_SEND" command received (if challenge-response is passed), encrypted configuration will be written to characteristic with index at the start. 'index' (zero) will not be encrypted but 'start_config' and 'stop_config' is encrypted. 'reserved' maybe filled with 0xFF or garbage.

```
typedef struct __attribute__((packed)) {
    int16_t records[15];
    records_crc_t crc;
} sRECORDS_Record_t;

typedef struct __attribute__((packed)) {
    uint16_t chunk_index;
    sRECORDS_Record_t record_structs[4];
} sRECORDS_Record_Chunk_t;
```

Figure 3. Structure of decrypted "record data" when index larger than 0 (zero).

Table 15. Example of encrypted and decrypted data of "record data *" (header).*

	Bytes												
	timestamp				interval		CRC		Record Count		CRC		Reserved (24 Bytes)
Received (hex)	D4	8E	04	2D	45	D6	0E	AD	11	82	29	9E	FFFF…
Deciphered (hex)	0E	B5	11	64	3C	00	9F	64	29	00	1F	90	FFFF…
Meaning	1678882062				60		0x9F64		41		0x1F90		-

Until "SEND_NEXT_CHUCK" command received and passed the challenge-response, data at the "Record Data" characteristic will be same. If secure "START_RECORD_SEND" received, data sending will be restart and data with zero index will be sent.

When "SEND_NEXT_CHUCK" command received successfully, with the start of 'index' 0, sending temperature records starts. Size of every chunk is 130 bytes. First 2 bytes is little-endian uint16_t 'chunk index' and remaining data structure contains 60 records and 4 CRC. After index there is 15 records and CRC. After that, another 15 records and CRC. Then another 15 records and CRC. Finally last 15 records and CRC.

Except chunk index, all elements of 'records' array is encrypted. Also, every member of record_t is little-endian.

Temperatures are stored in int16_t as multiplied by 100. For example, 23.42 Celsius is stored as 2342. If recording stopped by user before third temperature measurement, "INT16_MIN"(-32768) will be written to remaining element and CRC will be calculated with those values. Remaining bytes of characteristic will be filled with 0xFF.

The last data of any page may contain less than 60 records, in this case remaining space will be filled with 0xFF. After all records downloaded from a page, next chunk will be page header. This will contain start timestamp, number of records and CRC. If the recording to this page is finished, the stop timestamp will be valid.

Tel. +370 5 212 7472
Fax. +370 5 276 1380
www.teltonika-gps.com
info@teltonika.lt

If there is no more data stored in flash, “SEND_NEXT_CHUCK” command will return error and data at the “Record Data” characteristic will not change.

Table 4:Example of encrypted and decrypted data of “record data “.

	Bytes													
	index		1th temp		2nd temp		3rd temp		…		60th temp		CRC	
Received (hex)	03	00	7F	8B	B8	3D	9D	3F	70	3F	7F	8B	D2	68
Deciphered (hex)	03	00	0C	09	0B	09	0E	09	CA	8D	0C	09	00	80
Meaning	3		2316		2315		2318		0x8DCA		2316		-32768	

Command

This characteristic is for sending records related commands and 1 response for that command. After every write, 1 byte response to that command can be read from same characteristic. Commands must be encrypted but response is not encrypted.

Every command must start with little-endian uint16_t random value that received from “Random Value” characteristic. Then, little-endian uint16_t command value must be present. Except “START_RECORDING”, total size of all commands are 4 Bytes.

Table 5:Command list can be sent via “Command “ characteristic.

Command Name	Value	Size of parameter	
START_RECORD	0x0001	6 Bytes	It starts the recording with the given interval. If recording is already stopped and send interval to start is the same measurement will be continued and stored in next page. If interval is different all records data is deleted.
STOP_RECORD	0x0002	0	Stops the ongoing recording.
DELETE_RECORD	0x0003	0	Deletes the stored records and recording state stays. If device was recording it continues recording but from the start.
START_RECORD_SEND	0x0004	0	Starts to send Record chunks.

SEND_NEXT_CHUNK	0x0005	0	Sends next chunk.
TIME_SYNC	0x0006	4 Bytes	Send UNIX time for syncing the device. With this received timestamp, the device will update its time. If necessary, it will create records or it will skip some records. Refer: *Figure 6*
START_RECORD_SEND_TS	0x0007	4 Bytes	Send from what UNIX time data must be sent.
SEND_CURRENT_TS	0x0008	0	This command will write current timestamp of the device to *Record Data* characteristic if the device is recording. Data will be 4 bytes little-endian, unsigned, and encrypted.
START_FAST_RECORD_DOWNLOAD	0x0009	0	Starts to send Record chunks. After this command sent, reading "*Record Data*" char is enough. After each read, next chunk will be written to same char automatically.

The received command will be decrypted by firmware and random value will be compared with value of "Random Value" characteristic.

```
struct {
   uint16_t interval;
   uint32_t timestamp;
} ;
```

Figure 4. Structure of parameter of "START_RECORDING" command.

Parameter of "START_RECORDING" should consist of little-endian interval in unit of seconds and Unix timestamp.

Table 6:An example for "START_RECORD" command

	Random Value	Command	interval	Unix timestamp
Decrypted Command (Hex)	C4-57	01-00	2C-01	01-FD-B7-62
Meaning	22468	0x0001	300	1656225025

Parameter of "TIME_SYNC" should consist of little-endian Unix timestamp.

Table 7:An example for "TIME_SYNC" command.

	Random Value	Command	Unix sync timestamp
Decrypted Command (Hex)	C4-57	06-00	01-FD-B7-62
Meaning	22468	0x0006	1656225025

Parameter of "START_RECORD_SEND_TS" should consist of little-endian Unix timestamp.

Table 8:An example for "TIME_ START_RECORD_SEND_TS" command.

	Random Value	Command	Unix sync timestamp
Decrypted Command (Hex)	C4-57	07-00	01-FD-B7-62
Meaning	22468	0x0007	1656225025

Other commands do not require any parameter.

When a command is sent, the result will be written by firmware to the same characteristic. Length of these responses is 1 byte.

Table 9:Command Responses.

Response Name	Value	Meaning
ERROR_SUCCESS	0x00	There is no error. Command received successfully
ERROR_GENERAL	0x01	Unspecified Error.
ERROR_DECRYPTION	0x02	Received command could not successfully decrypted.
ERROR_RANDOM_VALUE	0x03	Received random value does not match with correct one.
ERROR_UNKNOWN_CMD	0x04	Received command is not known.
ERROR_LENGTH	0x05	Length of received command is wrong.
ERROR_NOT_STARTED	0x06	Recording did not start because of invalid parameter or device is already a recording.
ERROR_NOT_STOPPED	0x07	Recording couldn't be stopped or Command couldn't be performed because recording still in session.
ERROR_NO_MORE_CHUNCK	0x08	No more data chunk to be send.
_ERROR_NO_DATA_TS	0x09	There is no data from asked timestamp.
ERROR_SENDING_NOT_STARTED	0x0A	Send next chuck command occurred before send first chuck or send from timestamp.
ERROR_NO_DATA	0x0B	There is no data to send.

Tel. +370 5 212 7472
Fax. +370 5 276 1380
www.teltonika-gps.com
info@teltonika.lt

Workflow of reading:

1. Read "*Record Info*" Char and check if '*number_of_records*' is zero (Table 15). If it is 0, do not proceed.
2. Send start reading command START_RECORD_SEND or START_RECORD_SEND_TS.
3. Read Command char if ERROR_NO_DATA or ERROR_NO_DATA_TS was not raised because if they are data would not be present.
4. Read header. Record count value has information how many records there is in page. Header packet will always have index 0.
5. Record count should be subtracted by received record count so the user would know how many packets will be send if after subtracting there is remain 1 should be added to packet number. So calculated packet number + 1 from packet index inside data packet.
6. Sending 0x0005 command for continuing reading from Record Data char. First data packet of the page will always be 0.
7. When all the records from that page have been read;
8. User must read Command char to check if it is not no more chunks left.
9. If flag is ERROR_NO_MORE_CHUNCK all data has been read.
10. If flag is not ERROR_NO_MORE_CHUNCK sending 0x0005 next packet will be header of the next page. Steps from 4 can be repeated.

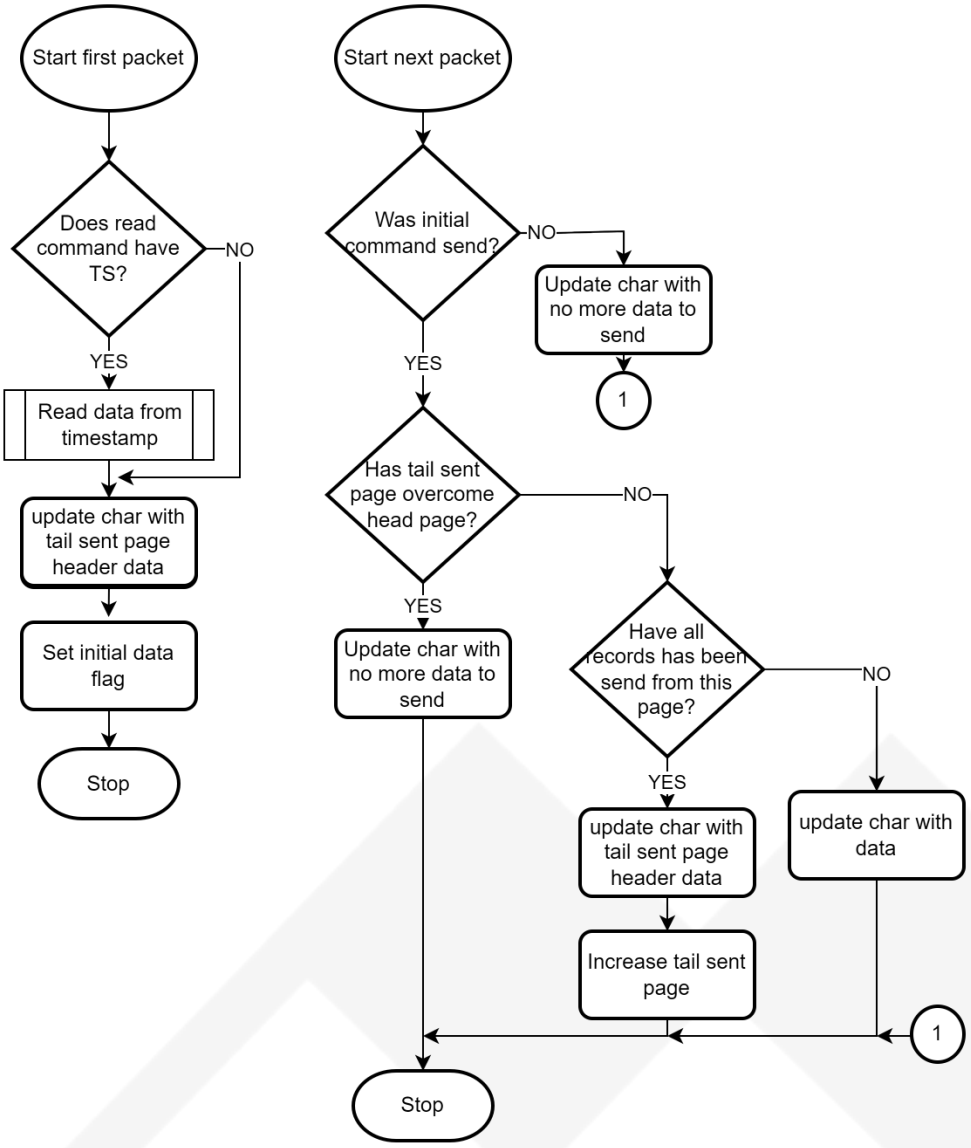

Figure 4. Reading algorithm

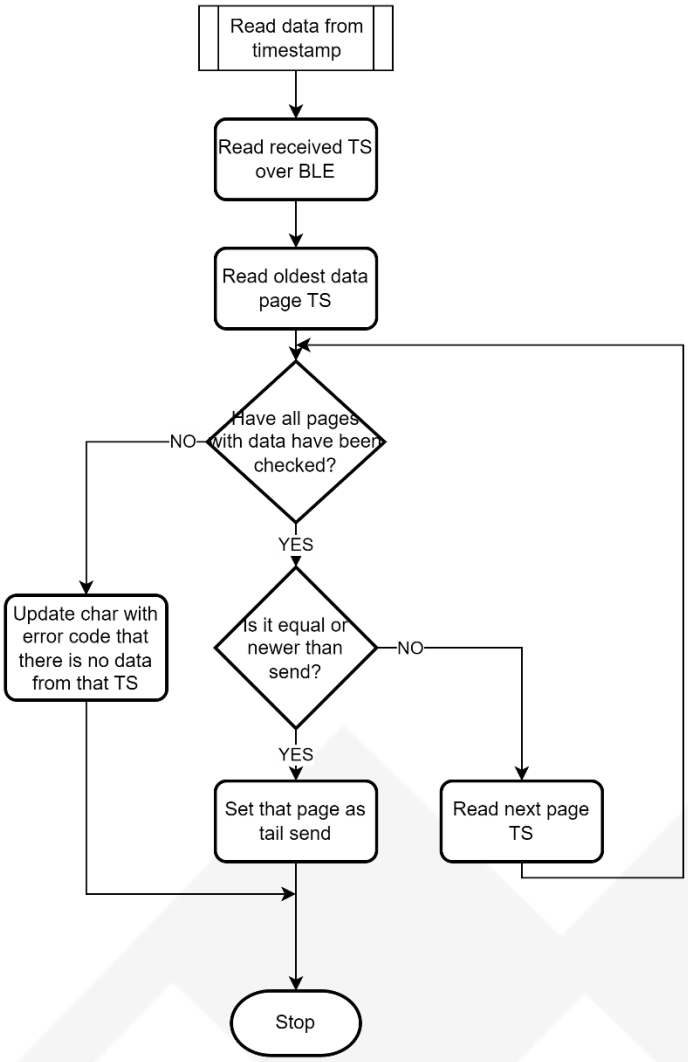

Figure 5. sent timestamp searching

Tel. +370 5 212 7472
Fax. +370 5 276 1380
www.teltonika-gps.com
info@teltonika.lt

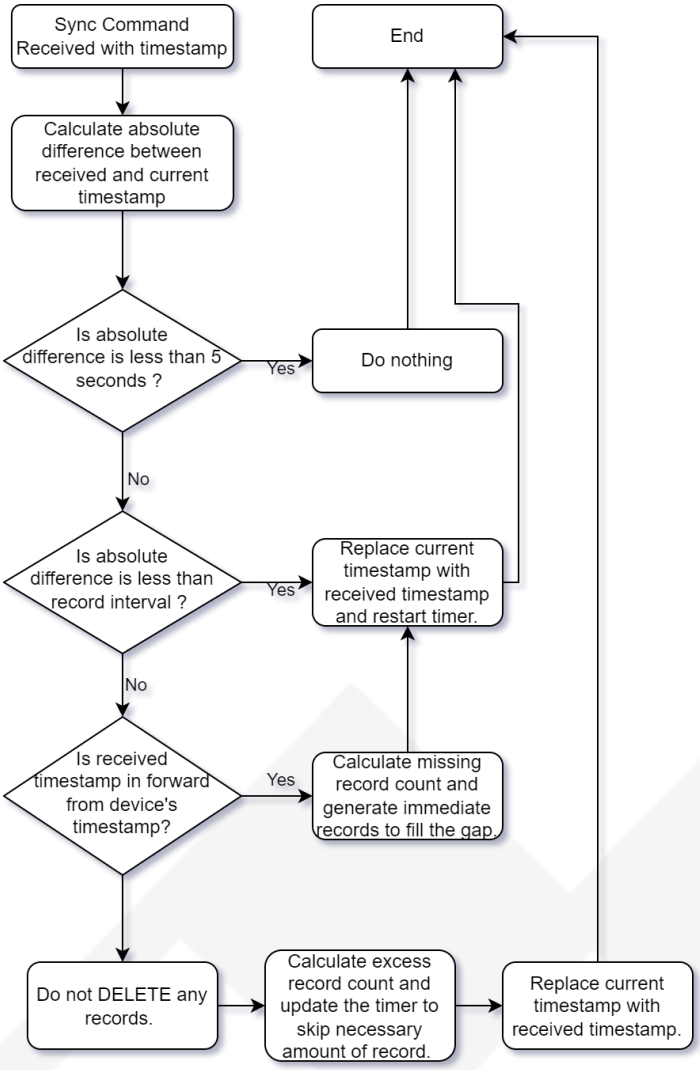

Figure 6: Algorithm for time sync

Tel. +370 5 212 7472
Fax. +370 5 276 1380

www.teltonika-gps.com
info@teltonika.lt

Encryption algorithm

The Tiny XTEA algorithm is used for this firmware. For detailed information, Refer to document about XTEA algorithm.

All commands, data received from the device uses this encryption method.

Changelog

Date	Author	Revision	Changes
2024.10.10	AR	0	Initial document created.
2025.06.05	OO	1	Add Record count checking to “Workflow of reading”. Changing the index of Table 5 to 15.