**TomTom Watch Interface**



Table of Contents

[TomTom Watch interface 3](#_Toc535655678)

[Introduction 3](#_Toc535655679)

[Interface 3](#_Toc535655680)

[File system 3](#_Toc535655681)

[Overview 3](#_Toc535655682)

[Preferences file, 0x00f20000 5](#_Toc535655683)

[Firmware files (0x000000F0, 0x00000012, 0x00010200) 6](#_Toc535655684)

[GPS Quickfix file (0x00010100) 7](#_Toc535655685)

[Manifest files (0x0085000n) 7](#_Toc535655686)

[Activity files (0x0091nnnn) 8](#_Toc535655687)

[Handling 8](#_Toc535655688)

[File format 8](#_Toc535655689)

[Header record 9](#_Toc535655690)

[Status record 10](#_Toc535655691)

[GPS Record 10](#_Toc535655692)

[Extended GPS record 11](#_Toc535655693)

[Heart rate record 11](#_Toc535655694)

[Summary record 12](#_Toc535655695)

[Pool size record 12](#_Toc535655696)

[Wheel size record 12](#_Toc535655697)

[Training setup record 13](#_Toc535655698)

[Lap record 13](#_Toc535655699)

[0x30 record 13](#_Toc535655700)

[Cycling cadence record 14](#_Toc535655701)

[Treadmill record 14](#_Toc535655702)

[Swim record 14](#_Toc535655703)

[0x37 record 15](#_Toc535655704)

[Interval setup record 15](#_Toc535655705)

[Interval start record 15](#_Toc535655706)

[Interval finish record 16](#_Toc535655707)

[Race setup record 16](#_Toc535655708)

[Race result record 16](#_Toc535655709)

[Altitude update record 17](#_Toc535655710)

[Heart rate recovery record 17](#_Toc535655711)

[Indoor cycling record 17](#_Toc535655712)

[Gym record 18](#_Toc535655713)

[Movement record 18](#_Toc535655714)

[Route description record 18](#_Toc535655715)

[Elevation record (Adventurer) 19](#_Toc535655716)

[Battery record 20](#_Toc535655717)

[Fitnesspoints record 21](#_Toc535655718)

[Workout record 1 21](#_Toc535655719)

[Workout record 2 21](#_Toc535655720)

[Activity tracking files (0x00b1nnnn, 0x00b2nnnn, 0x00b3nnnn) 22](#_Toc535655721)

[Routefiles (0x00b8nnnn) 25](#_Toc535655722)

[Activity types 27](#_Toc535655723)

[USB Interface 28](#_Toc535655724)

[Generic mechanism 28](#_Toc535655725)

[Higher level functions 30](#_Toc535655726)

[Open File/Close File/Delete file 30](#_Toc535655727)

[Reading files 31](#_Toc535655728)

[Writing files 32](#_Toc535655729)

[Request file size 33](#_Toc535655730)

[Listing/enumerating files 34](#_Toc535655731)

[Get Watch time 35](#_Toc535655732)

[Firmware version (MSG\_GET\_FIRMWARE\_VERSION) 35](#_Toc535655733)

[Bluetooth Low Energy version (MSG\_GET\_BLE\_VERSION) 36](#_Toc535655734)

[Product ID (MSG\_GET\_PRODUCT\_ID) 36](#_Toc535655735)

[Reset device (MSG\_RESET\_DEVICE) 36](#_Toc535655736)

[Reset GPS processor (MSG\_RESET\_GPS\_PROCESSOR) 37](#_Toc535655737)

[Format Watch (MSG\_FORMAT\_WATCH) 38](#_Toc535655738)

# TomTom Watch interface

## Introduction

The TomTom Watch family (Multisport, Spark, Runner, Adventurer) are GPS watches with a multitude of fitness tracking functions. TomTom still adds new functions, though (07-2017) it appears that TomTom will move out of wearables. The watches are used in combination with the TomTom Mysports cloud account. A local application on PC or Mobile/Tablet is used to communicate to the watch and sync between the watch and the cloud.

In order to create a PC application I continued reverse engineering the watch. The start of this was formed by the excellent application ttwatch of Ryan Binns. It has been applied in the TomTomWatch Application (<http://blog.studioblueplanet.net/?page_id=566>).

‘not checked’ indicates I copied the information from the Ryan Binns application, but did not check myself.

## Interface

The watch has two physical interfaces:

* Bluetooth
* USB

This document is limited to the USB interface. Using this interface a client can issue commands to the watch and read, write and delete files. First the file system and formats are described, then the USB interface.

# File system

## Overview

The watch interface functions are based on a file system that is accessible via Bluetooth and the USB interface. Note that another file system is present that is accessible via the PC via USB as regular removable storage and used for transfer of music. This file system is mapped on the PC file system (like an USB stick) when the watch is connected.

This section describes the former filesystem. Files are addressed by an 32 bit integer ID. In this document the file IDs are given in an 8 character hexadecimal, like **0x00910000**.

Following files have been identified:

|  |  |
| --- | --- |
| File ID | Description |
| 0x00000012 | Bluetooth Low Energy (BLE) firmware. Used for firmware upgrades. not checked |
| 0x000000F0 | System firmware. Used for firmware upgrades |
| 0x00010100 | GPS Quickfix data, used for obtaining a quick GPS lock. Is written each time the watch is connected to TomTom MySports. For the adventurer, the file is downloaded from <https://gpsquickfix.services.tomtom.com/fitness/sifgps.f2p3enc.ee> |
| 0x00010200 | GPS Firmware. not checked |
| 0x00010301 | Some version numbers?? |
| 0x00013000 | Stacktrace. Text file |
| 0x00013001 | BLE firmware Update log not checked |
| 0x00013002 | Sysem firmware Update log. Text file |
| 0x00013100 | System log. Text file |
| 0x0071xxnn | Races. xx defines the activity. nn is the race number. Proprietary format. |
| 0x0072nnnn | Race history. not checked |
| 0x0073xxnn | History data. For each activity type (xx) for the last 10 activities (0xnn) such a file is generated |
| 0x0081nnnn | Language files |
| 0x008300xx | Activity summary. Each file contains the last 10 activities. This file is used by the watch to show the last 10 activities for each activity type xx. |
| 0x0085000n | Manifest files. Contain the settings. 0x00850000 is the current list of settings. 0x00850001 and 0x00850002 are backups |
| 0x00880000 | Textual representation of the playlists (music watches only) |
| 0x0091nnnn | Activity files. Correspond with the ttbin files. Proprietary file format containing records. |
| 0x00B100nn | Tracked activity (steps, calories, heart rate, sleep, fitness points, etc) for upload to TomTom MySports. Up to 20 files are generated. A new file is generated each time the watch is connected to the PC and disconnected. The files are uploaded and deleted when the watch is connected to TomTom MySports. Protobuf format |
| 0x00B20000 | Tracked activity, temporary file which is used during the time the watch is connected to the PC. When disconnected, this file is ‘renamed’ to the next 0x00B100nn file (if there are 20 0x00B100nn files, tracked activity keeps being logged to this file; rename after the the 0x00B100nn files have been deleted; CHECK!). Same protobuf format as 0x00B100nn |
| 0x00B20001 | Unclear. 4 bytes. |
| 0x00B3000n | Tracked activity of the last 7 days. n=8-f. Same protobuf format as 0x00b100nn, however heartrates are not stored. |
| 0x00B8000n | Routes (track planning). Each file contains a route. Protobuf format. The watch accepts 15 route files, hence 0x0≤n≤0xE |
| 0x00B9nnnn |  |
| 0x00BEnnnn | The personalized workouts, added since firmware version 1.7.53 (Adventurer). Protobuf format. |
| 0x00F20000 | Preferences file. XML format. Contains the watch name and other preferences for connecting to TomTom MySports |

## Preferences file, 0x00f20000

The preferences file is an XML file containing the preferences used when connecting to TomTom MySports. After factory reset the watch does not have such a preference file. When connecting the watch to the ‘TomTom Sports Connect’ it is created as part of the registration process. A.o. a token and secret are generated and coupled to a TomTom Mysports cloud account. When connecting the Watch to ‘TomTomWatch’ a default preference file is written without token and secret (file will be overwritten by ‘TomTom Sports Connect’).

Apart from the token and secret it contains the watch name, configuration service URL and some additional configuration. The file is also copied on a Windows PC to:

c:\users\<user>\TomTom Sports\<device serial>\preferences.dat

Changing the watch name can be done by changing the name between the <watchName></watchName> tags.

<?xml version="1.0" encoding="UTF-8"?>

<preferences version="1" modified="Sun Oct 29 17:12:36 2017">

<ephemerisModified>0</ephemerisModified>

<watchName>GPS Watch Jorgen</watchName>

<ConfigURL>https://mysports.tomtom.com/service/config/config.json</ConfigURL>

<exporters>

<online>

<export id="MySports" autoOpen="1"/>

<MySportsAuthToken>...</MySportsAuthToken>

<MySportsTokenSecret>...</MySportsTokenSecret>

</online>

</exporters>

</preferences>

## Firmware files (0x000000F0, 0x00000012, 0x00010200)

Firmware update steps are:

1. Fetch the configuration service as defined in the preferences file, <ConfigURL> tags.  
   E.g.   
   <ConfigURL>https://mysports.tomtom.com/service/config/config.json</ConfigURL>
2. Get the “service:firmware” url from the resulting JSON:  
   E.g.   
   https://sports.tomtom-static.com/downloads/firmware/{PRODUCT\_ID}/FirmwareVersionConfigV2.xml
3. Fill in the {PRODUCT\_ID} as resulted from the product ID USB function.  
   E.g. for the adventrurer:   
   https://sports.tomtom-static.com/downloads/firmware/E0070000/FirmwareVersionConfigV2.xml
4. Retrieve the xml file. This xml file defines the latest firmware and corresponding firmware files

<FirmwareVersion>

<latestVersion>

<Major>1</Major>

<Minor>7</Minor>

<Build>53</Build>

</latestVersion>

<isCritical>yes</isCritical>

<URL>1\_7\_53/0x000000F0</URL>

</FirmwareVersion>

1. Download the firmware files from the same location  
   E.g.   
   https://sports.tomtom-static.com/downloads/firmware/E0070000/1\_7\_53/0x000000F0
2. Upload the firmware files to the watch
3. Execute an USB reboot command to the watch.

Note:

* Firmware upgrades usually lead to an extension of the number of settings as stored in the manifest files (0x0085000n).
* I downgraded the firmware, by applying 1.3.255 0x000000F0 file to my watch running 1.6.26. This succeeded, however it resulted in a full reset and a required a reconfiguration. Not clear if downgrading is generally supported.

## GPS Quickfix file (0x00010100)

Procedure for uploading quickfix data:

1. Fetch the configuration service as defined in the preferences file, <ConfigURL> tags.  
   E.g.   
   <ConfigURL>https://mysports.tomtom.com/service/config/config.json</ConfigURL>
2. Get the “service:ephemeris” url from the resulting JSON:  
   E.g.   
   https://gpsquickfix.services.tomtom.com/fitness/sifgps.f2p{DAYS}enc.ee
3. Replace the {DAYS} by the number of days ahead. 3 and 7 seem the only possible values…  
   E.g.   
   https://gpsquickfix.services.tomtom.com/fitness/sifgps.f2p3enc.ee
4. Download the file.
5. Upload the file to the watch using file ID 0x00010100.

## Manifest files (0x0085000n)

These files contain the watch settings as key-value pairs. The number of settings is defined by the Length field. The list is settings is extended usually at software updates.

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| File type | 0x0085 | 2 | Integer |
| Length | Number of tag-value pairs | 2 | Integer |
| Array: |  |  |  |
| Tag | Tag. Seems to be an increasing number from 0 to (Length-1) | 2 | Integer |
| Value | Value | 4 | Integer |

## Activity files (0x0091nnnn)

### Handling

When using the TomTom Sports Connect application, the 0x0091NNNN files are downloaded and stored on disk. Under windows the path is:

c:\users\<user>\TomTom Sports\<watch name>\<YYYY-MM-DD>\<sport>\_<hh-mm-ss>.ttbin

For example:

c:\users\<user>\TomTom Sports\GPS Watch Jorgen\2017-01-01\Freestyle\_20-35-58.ttbin

When using the Android app the files are stored on the file store:

/TomTom\_MySports/<serial>/workouts/uploaded/<fileId>\_<YYYYMMDD>\_<hhmmss>.ttbin

For example:

/TomTom\_MySports/HL1456G01770/workouts/uploaded/00910000\_20170101\_203558.ttbin

(Writing a ttbin file to /TomTom\_MySports/<serial>/workouts/ and starting the TomTom App makes the App upload the ttbin file to the TomTom MySports cloud.

### File format

Activity files contain the logged activities. After deleting all **0x0091nnnn** files, the next activity is logged with nnnn=0, subsequent files are logged by increasing nnnn by 1.

Format:

|  |
| --- |
| Record 0 – Header |
| Record 1 |
| Record 2 |
| … |
| Record N |

The ttbin file consist of a series of record. Each record starts with a tag followed by a number of values. The tag identifies the record type and defines the values to follow. A value can consist of 1 or more bytes encoding an integer or float value.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tag | Value 1 | Value 2 | Value 3 | … | … | Value M |

All integers are little endian (LSB first)

The first record in the file is the header record (tag=0x20). This is a special record a.o. defining the records in the file.

### Header record

It is the first record in the ttbin file. The header defines the ttbin file. Amongst others it defined the record types that occur in the file with their lengths.

Length: 117 (version<=0x09) or 120 bytes (version>=0x0a), excluding the array with tags and lengths.

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x20 | 1 | integer |
| version | Version of the ttbin file format. Versions of 0x07, 0x09 and 0x0A have been checked. | 1 | integer |
| firmware version | Versions of  watch firmware, consisting of major, medium, minor, like 1.3.255.  In ttbin “version” <=0x09 for each part 1 byte is reserved, “version” >=0x0A two bytes, little endian. | Version <=0x09: 3  >=0x0A: 6 | integer |
| product ID | ID of the product. For Adventurer: 0xE0070000, Runner 3 Music 0xD1070000 | 2 | integer |
| Start time | Start time of the activity, as epoch seconds, e.g. 0x5A0EF328 which corresponds to GMT: Friday, November 17, 2017 14:33:12 | 4 | integer |
| Software version | On the Adventurer: all 0x00 | 16 | byte array |
| GPS firmware version | On the Adventurer: all 0x00 | 80 | byte array |
| Watch time | Watch time as epoch seconds. On the adventurer the same as “Start time” | 4 | integer |
| Local time offset | Time offset between local time and GMT. For Amsterdam this is 3600 seconds in winter, 7200 seconds in summer. | 4 | integer |
| Reserved |  | 1 |  |
| Length records | The next section of the header defines the record tags that appear in the file, with the corresponding record length. This field defines the number of tag-length pairs in the array. | 1 | integer |
| Array: |  |  |  |
| Tag | Record tag, defining the type of record | 1 | integer |
| Length | Length of the record in bytes. For a number of records (e.g. 0x4B) a length of 0xFFFF is defined, meaning variable length. In that case the length is defined in the record itself. | 2 | integer |

### Status record

The record indicates status changes. READY -> ACTIVE <-> PAUSED -> STOPPED. READY is the state when the activity is chosen. ACTIVE is when ‘get going’ is selected. PAUSED when the activity is PAUSED by pressing the left button on the watch. STOPPED is when the left button is pressed another time and the activity is finished.

Length: 7 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x21 | 1 | Integer |
| Status | New status: READY – 0, ACTIVE – 1, PAUSED – 2, STOPPED – 3 | 1 | Integer |
| Activity | Activity code | 1 | Integer |
| Timestamp | GMT Timestamp in epoch seconds | 4 | Integer |

### GPS Record

This record is added each second when the watch is in the ACTIVE state.

Length: 28 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x22 | 1 | Integer |
| Latitude | Latitude \* 1E7 degrees, -180E7 – 180E7 degrees | 4 | Integer |
| Longitude | Longitude \* 1E7 degrees, -180E7 – 180E7 | 4 | Integer |
| Heading | Heading \* 1E2 degrees, 0-360E2 | 2 | Integer |
| Speed | Speed in 1E2 m/s | 2 | Integer |
| Timestamp | GMT Timestamp in epoch seconds | 4 | Integer |
| Calories | Cumulative calories burned (cal) | 2 | Integer |
| Filtered speed | Some filtered speed value in m/s | 4 | Float |
| Distance | Cumulative distance in m | 4 | Float |
| Cycles | The cycles per second. For running 2-3. | 1 | Integer |

### Extended GPS record

Additional information regarding the GPS tracking.

Length: 20 (version<=0x09) or 24 bytes (version>=0x0a).

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x23 | 1 | Integer |
| EVPE | Estimated Vertical Precision Error in cm | 2 | Integer |
| EHPE | Estimated Horizonal Precision Error in cm | 2 | Integer |
| HDOP | Horizontal Dilution of Precision | 1 | Integer |
| Unknown |  | 4 | Int Array |
| Unknown |  | 4 | Int Array |
| Unknown |  | 4 | Int Array |
| Unknown |  | 1 | Integer |
| Unknown |  | 1 | Integer |
| TBD |  | 4 |  |

### Heart rate record

This record is added each second when the watch is in the ACTIVE state and the HR sensor is active or an external HR sensor is connected.

Length: 6 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x25 | 1 | Integer |
| Unknown | 0xFF for external HR sensor, other value for internal sensor | 1 | Integer |
| Timestamp | Timestamp in epoch seconds. Oddly enough, this is local time, whereas the rest of the timestamps in is GMT. | 4 | Integer |

### Summary record

Summary of the activity. Logged when the activity is STOPPED.

Length: 14 (version<=0x09) or 18 bytes (version>=0x0a).

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x27 | 1 | Integer |
| Activity | Activity code | 1 | Integer |
| Distance | Distance | 4 | Integer |
| Duration | Duration of the activity in seconds. Excluding pause. | 4 | Integer |
| Calories | Calories burned during the activity | 2 | Integer |
| Unknown | ? 0x004A=74 – Starting heartrate?? | 2 | Integer |
| Duration2 | Seems to be the duration. When paused slightly longer (2-3 sec) than Duration… | 4 | Integer |

### Pool size record

Pool size. Used in Swimming activity.

Length: 5 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x2A | 1 | Integer |
| Pool size | Pool size in cm | 4 | Integer |

### Wheel size record

Wheel size. Used in cycling.

Length: 5 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x2B | 1 | Integer |
| Wheel size | Wheel circumference in mm, as defined under Cycling. | 4 | Integer |

### Training setup record

Defines the training. Not used if no training set.

Length: 10 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x2D | 1 | Integer |
| Goal | Training goal  0 = goal distance, 1 = goal time, 2 = goal calories, 3 = zones pace, 4 = zones heart, 5 = zones cadence, 6 = race, 7 = laps time, 8 = laps distance, 9 = laps | 1 | Integer |
| Minimum | Minimum value: metres, seconds, calories, sec/km, km/h, bpm sec/km, km/h, bpm (only used for zones). | 4 | Float |
| Maximum | Maximum value. Used in combination with the miminum, e.g. to indicate a heartrate zone min and max value. If only one limit is needed, only minimum is used and Maximum is set to 0x00000000. | 4 | Float |

### Lap record

Lap not checked

Length 11 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x2F | 1 | Integer |
| Time | Total time in seconds | 4 | Integer |
| Distance | Total distance in meters | 4 | Float |
| Calories | Total calories (cal) | 2 | Integer |

### Time to satellite fix record

This record shows the time it took to get a satellite fix. The record is only written once at the start of the file, even if it repeatedly gets and loses a fix.

Length 3 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x30 | 1 | Integer |
| Time to fix | The time it takes for the watch to get ready, i.e. to have a satellite fix, in seconds. During this period the watch displayes ‘Please wait’. To be precise: it appears to be the time between selecting the activity and the watch showing the ‘GET GOING’ message minus 2 seconds. | 2 | Integer |

### Cycling cadence record

Revolutions and time counters. Can be used to calculate the cadence. not checked

Length 11 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x31 | 1 | Integer |
| Wheel revolutions | Counts the wheel revolutions | 4 | Integer |
| Wheel revolutions time | Counts the time in ms | 2 | Integer |
| Crank revolutions | Counts the crank revolutions | 2 | Integer |
| Crank revolutions time | Counts the time in ms | 2 | Integer |

### Treadmill record

Treadmill. not checked

Length: 17 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x32 | 1 | Integer |
| Timestamp | Timestamp in epoch seconds, UTC | 4 | Integer |
| Distance | Total distance in m | 4 | Float |
| Calories | Calories burned | 2 | Integer |
| Steps | Number of steps since ?? | 4 | Integer |
| Step length | Step length in cm | 2 | Integer |

### Swim record

Treadmill. not checked

Length: 21 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x34 | 1 | Integer |
| Timestamp | Timestamp in epoch seconds, UTC | 4 | Integer |
| Distance | Total distance in m | 4 | Float |
| Frequency |  | 1 | Integer |
| Stroke type |  | 1 | Integer |
| Strokes | Strokes since the last record | 4 | Integer |
| Completed laps |  | 4 | Integer |
| Calories |  | 2 | Integer |

### 0x37 record

Occurs a few seconds after the watch is set to active (activity is started). not checked

Length: 2 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x37 | 1 | Integer |
| ? | Counter or status value?? Value: 1. No lap counter, no intervals counter. | 1 |  |

### Interval setup record

Interval training setup as defined on the watch. not checked

Length: 22 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x39 | 1 | Integer |
| Warm type | 0 – Distance, 1 - Time | 1 | Integer |
| Warm | Warm up in meters or seconds | 4 | Integer |
| Work type | 0 – Distance, 1 – Time | 1 | Integer |
| Work | Work in meters or seconds | 4 | Integer |
| Rest type | 0 – Distance, 1 – Time | 1 | Integer |
| Rest | Rest in meters or seconds | 4 | Integer |
| Cool type | 0 – Distance, 1 – Time | 1 | Integer |
| Cool | Cool down in meters or seconds | 4 | Integer |
| Sets | Number of sets | 1 | Integer |

### Interval start record

Start of the interval. not checked

Length: 2 bytes (version ≤ 0x09), 3 bytes (version ≥ 0x0a)

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x3A | 1 | Integer |
| Type | 1 - warm up, 2 - work, 3 - rest, 4 - cool down, 5 - finished | 1 | Integer |

### Interval finish record

Interval finish report. not checked

Length: 12 bytes (version ≤ 0x09), 14 bytes (version ≥ 0x0A)

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x3B | 1 | Integer |
| Type | 1 - warm up, 2 - work, 3 - rest, 4 - cool down, 5 - finished | 1 | Integer |
| Time | Duration of the interval in seconds | 4 | Float |
| Distance | Distance covered during the interval in m | 4 | Integer |
| Calories | Calories burned | 2 | Integer |
| ? |  |  |  |

### Race setup record

Race definition file. not checked

Length: 41 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x3C | 1 | Integer |
| Race ID | Only used for web services race, otherwise 0 | 16 | Byte array |
| Distance | Distance in m | 4 | Float |
| Duration | Duration in seconds | 4 | Integer |
| Name | Null terminated character string | 16 | Char array |

### Race result record

Race results. not checked

Length: 11 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x3D | 1 | Integer |
| Distance | Distance in m | 4 | Float |
| Duration | Duration in seconds | 4 | Integer |
| Calories | Calories burned | 2 | Integer |

### Altitude update record

Altitude record. Since version ≥ 0x0A no longer present on the Adventurer. It has been replaced by the elevation record (tag=0x47) not checked

Length: 8 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x3E | 1 | Integer |
| Rel. Altitude | Relative altitude since start of the workout | 2 | Integer |
| Climb | Total climb | 4 | Float |
| Qualifier | Not defined yet | 1 |  |

### Heart rate recovery record

This record presents the heart rate recovery. The recovery is measured when the watch is set to pause. During 1 minute the decrease in heart rate is recorded.

Length: 9 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x3F | 1 | Integer |
| Score | Score: 0 – no recovery, 1 – poor recovery, 2 – fair recovery, 3 – good recovery, 4 – excellent recovery (≥40 bpm) | 4 | Integer |
| Recovery | Heart rate recovery in BPM per minute. A positive value means decrease, a negative value means an increase after the minute (no recovery). | 4 | Integer |

### Indoor cycling record

Indoor cycling not checked

Length: 12 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x40 | 1 | Integer |
| Timestamp | Timestamp in epoch seconds, UTC | 4 | Integer |
| Distance | Distance in m | 4 | Integer |
| Calories | Calories burned | 2 | Integer |
| Cadence | Cadence | 1? | Integer |

### Gym record

Gym record not checked

Length: 11 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x41 | 1 | Integer |
| Timestamp | Timestamp in epoch seconds, UTC | 4 | Integer |
| Calories | Calories burned | 2 | Integer |
| Cycles | Total number of cycles | 4 | Integer |

### Movement record

Some status regarding movement TBD

Length: 2 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x42 | 1 | Integer |
| Movement status | Not clear: 0 - standing still, 1 – moving slower, 2 – moving, 3 – moving?? |  |  |

### Route description record

Description of the planned route.

Length: 101 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x43 | 1 | Integer |
| ?? | 0x00? | 16 |  |
| ?? |  | 4 |  |
| Route name | Null terminated string | 80 | Char array |

### Elevation record (Adventurer)

Contains barometric elevation information

Length: 12 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x47 | 1 | Integer |
| Unknown | Some status. Bit values. | 1 | Integer |
| Elevation 1 | Absolute altitude, probably GPS altitude, in m | 2 | Integer |
| Elevation 2 | Relative altitude, starting at 0, in m | 2 | Integer |
| Ascend | Total cumulative ascend in m | 2 | Integer |
| Descend | Total cumulative descend in m | 2 | Integer |
| Unknown | Seems to be a measure for the height increase (dz/dt) | 2 | Integer |

### Battery record

Battery record. Occurs at 30 s (occasionally at 60 s) intervals.

Figure 1 Typical battery level values (recorded during 1h20' run on Adventurer)

Length 5 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x49 | 1 | Integer |
| Level | Battery level in % (0-100). Tends to decreasing during the exercise as one might expect, though subsequent values might fluctuate (down and up). | 1 | Integer |
| Unknown1 | Typical value 127 | 1 | Integer |
| Unknown2 | Typical values 4, 5, 6 | 1 | Integer |
| Unknown3 | Typical value 0 | 1 | Integer |

### Fitnesspoints record

Contains the TomTom fitness points. Fitness points are an evaluation of the workout that depends on the heart rate.

Length 9 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x4A | 1 | Integer |
| Timestamp | Timestamp in epoch seconds | 4 | Integer |
| Fitnesspoints 1 | Cumulative fitness points | 2 | Integer |
| Fitnesspoints 2 | Appears to be same value as Fitnesspoints 1 | 2 | Integer |

### Workout record 1

Has something to do with the workout.

Length: variable (0xFF)

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x4B | 1 | Integer |
| Length | Length of the remainder of the record in bytes | 2 | Integer |
| ?? |  |  |  |
| ?? |  |  |  |

### Workout record 2

Has something to do with the workout. Contains the messages. The record does not seem to be mentioned in the header (tag=0x20)

Length: variable (0xFF)

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Bytes | Format |
| Tag | 0x4C | 1 | Integer |
| Length | Length of the remainder of the record in bytes | 2 | Integer |
| Header 0 | 5 Bytes header, the 5th byte is the length of the string | 21 |  |
| String 0 | String | 21+5 |  |
| Header 1 |  |  |  |
| String 1 |  |  |  |
| … |  |  |  |
| Header x |  |  |  |
| ?? |  |  |  |

## Activity tracking files (0x00b1nnnn, 0x00b2nnnn, 0x00b3nnnn)

Daily activity is tracked in the activity tracking files. The format is protobuf format. Next protobuf definition shows the content of the files.

|  |
| --- |
| //#################################################################################################  //  // Definition file for the personal workouts (0x00B1nnnn, 0x00B2nnnn, 0x00B3nnnn) files  //  //#################################################################################################  syntax = "proto2";  package tutorial;  *<Language specific statements>*  //#################################################################################################  // Root message  //#################################################################################################  message Root  {  repeated RootContainer rootContainer =1;  }  //#################################################################################################  // 1st level: root container  //#################################################################################################  message RootContainer  {  optional Metadata metadata =7;  optional DataContainer dataContainer =8;  }  //#################################################################################################  // 2nd level: metadata. Containing just two ints having the same value. Occurs often in one file  //#################################################################################################  message Metadata  {  required fixed32 manufacturer =1; // always 0x1234DAEB  required fixed32 fileType =2; // 0x00010100 for 0x00BE0000  }  //#################################################################################################  // 2nd level: data container. Contains various types of data  //#################################################################################################  message DataContainer  {  required SubDataContainer subDataContainer =1;  }  //#################################################################################################  // 3rd level: sub data container  //#################################################################################################  message SubDataContainer  {  optional DeviceInfo deviceInfo =1; // Device info  optional TrackRecord trackRecord =2; // activity tracking (calories, steps, hr, sleep, etc)  optional Record2 record2 =3; // ?  optional Record6 record6 =5; // ?  optional HeartRecord heartRecord =6; // Heart rate  optional Record4 record4 =7; // ?  optional FitnessRecord fitnessRecord =9; // TomTom FitnessPoints counters  }  //#################################################################################################  // 4th level: Device info  //#################################################################################################  message DeviceInfo  {  required string deviceName =1;  required fixed32 year =2;  required SoftwareVersion softwareVersion =3;  }  message SoftwareVersion  {  required int32 majorVersion =1;  required int32 mediumVersion =2;  required int32 minorVersion =3;  required Unknown01 unknown01 =4;  optional Unknown02 unknown02 =5; // 1.6.26  }  message Unknown01  {  optional int32 dummy =1;  }  message Unknown02  {  optional int32 dummy =1;  }  //#################################################################################################  // 4th level: Tracker Record  //#################################################################################################  message TrackRecord  {  required int32 recordId =1;  required int32 time =2; // Epoch time in seconds, 900 s interval = 15 min  required int32 timeZone =3; // Difference between time and UTC in seconds (?)  required int32 interval =4; // Interval of this measurement = 900s often  required int32 steps =5; // steps  required int32 active =6; // active time in seconds  required int32 distance =7; // distance in m  required int32 kcal =8; // kcal  required int32 kcalRest =9; // kcal in rest? (~12 kcal/10 min)  required int32 unknown1 =11; // 0  optional int32 sleepTime =12; // sleep time in seconds.  optional int32 sleepMode =13; // 1: active, 2: charging? 3: 1st hour sleep 4: sleep  // Total sleep time: sum sleep time if sleepMode=3 or 4  // from 12:00:00 till 12:00:00  }  //#################################################################################################  // 4th level: Data Record 2  //#################################################################################################  message Record2  {  required fixed32 time =1; // Epoch time in seconds, 900 s interval = 15 min  required int32 interval =2; // Interval in seconds? 14400=4h  repeated TagValueContainer tagValue =3;  required int32 unknown1 =4;  required int32 unknown2 =5;  optional int32 unknown3 =6;  optional fixed32 unknown4 =104; // 1.6.26  }  message TagValueContainer  {  required int32 tag =1;  required ValueContainer valueContainer =2;  }  message ValueContainer  {  required int32 value =3;  }  //#################################################################################################  // 4th level: Data Record 3 - heart rate  //#################################################################################################  message HeartRecord  {  required fixed32 time =1; // Epoch time in seconds, 900 s interval = 15 min  required int32 interval =2; // Interval in seconds?? 14400=4h  required int32 heartRate =3; // Heartrate in bpm. This value is displayed by TomTom MySports  required int32 value01 =4; // Some other heartrate value  required int32 value02 =5; // ?  }  //#################################################################################################  // 4th level: Data Record 4  //#################################################################################################  message Record4  {  required fixed32 time =1; // Epoch time in seconds, 900 s interval = 15 min  required int32 interval =2; // Interval in seconds?? 14400=4h or 14280=3h58'???  repeated int32 value =3; // 6 values in the message  }  //#################################################################################################  // 4th level: Data Record 5  //#################################################################################################  // Occurs every 120 seconds when logging activity  message UserData  {  required string version =1;  required int32 unknown1 =2;  required int32 unknown2 =3;  required int32 unknown3 =4;  required int32 length =5; // Users heigth in cm  required int32 weight =6; // User weight  required int32 unknown4 =7;  required int32 unknown5 =8;  required int32 unknown6 =9;  required int32 unknown7 =10;  required int32 unknown8 =11;  required int32 unknown9 =12;  required int32 unknown10 =13;  required int32 unknown13 =16;  required int32 unknown14 =17;  required int32 unknown15 =18;  required int32 unknown16 =19;  }  message FitnessRecord  {  required int32 time =2; // Epoch time in seconds, 120 s interval = 2 min during activity  required int32 interval =3; // Interval in seconds?? 14280=238 min = 3h58'??  optional UserData userData =5; // User data  optional int32 unknown1 =7;  optional int32 unknown2 =8;  optional int32 unknown3 =10;  optional int32 unknown4 =11;  optional int32 unknown5 =12;  required int32 fitnessPoints1 =15; // Cummulative TomTom activity points counter  required int32 fitnessPoints2 =16; // Cummulative TomTom activity points counter  }  //#################################################################################################  // 4th level: Data Record 6  //#################################################################################################  message Record6Sub  {  required int32 unknown1 =1;  required int32 unknown2 =2;  required int32 unknown3 =3;  }  message Record6  {  required int32 time =1; // Epoch time in seconds, 900 s interval = 15 min  required int32 interval =2; // Interval in seconds?? 14280=238 min = 3h58'??  required int32 unknown1 =3;  repeated Record6Sub sub =4; // Some data  } |

## Routefiles (0x00b8nnnn)

Route files were added on the TomTom Adventurer for ahead track planning. The files are Google ‘Protocol Buffers’ encoded. Protocol buffers (<https://developers.google.com/protocol-buffers/>: *Protocol buffers are Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data – think XML, but smaller, faster, and simpler*). The file format is defined in a .proto file that can be compiled into encoding/decoding code for various programming languages.



Below you find the .proto definition file for the route files.

|  |
| --- |
| //#################################################################################################  //  // Definition file for the TomTom routes (files with ID 0x00b8nnnn). A route is the result of  // trackplanning by converting and uploading GPX track files.  //  //#################################################################################################  syntax = "proto2";  package tutorial;  *<Language specific statements>*  //#################################################################################################  // ROOT level  //#################################################################################################  message Root  {  repeated RootContainer SubDataContainer =1;  }  //#################################################################################################  // 1st level: root container  //#################################################################################################  message RootContainer  {  optional Metadata metadata =7;  optional DataContainer dataContainer =8;  }  //#################################################################################################  // 2nd level: metadata. Containing just two ints having the same value. Occurs often in one file  //#################################################################################################  message Metadata  {  required fixed32 manufacturer =1; // always 0x1234DAEB  required fixed32 fileType =2; // 0x00070100 for route files  }  //#################################################################################################  // 2nd level: data container. Contains various types of data  //#################################################################################################  message DataContainer  {  required SubDataContainer subDataContainer =1;  }  //#################################################################################################  // 3rd level: sub data container  //#################################################################################################  message SubDataContainer  {  required TrackMetaData trackMetadata =1;  repeated StartCoordinate coordinate =2;  required SegmentData data =3;  }  //#################################################################################################  // 4th level: Some information on the route  //#################################################################################################  message TrackMetaData  {  required string name =1;  required BoundingBox box =2;  required bytes time =3;  }  //#################################################################################################  // 4th level: Start coordinate  //#################################################################################################  message StartCoordinate  {  required Coordinate coordinate =1;  required int32 index =2;  }  //#################################################################################################  // 4th level: Segment section  //#################################################################################################  message SegmentData  {  required int32 numberOfSegments =1;  repeated Segment data =2;  }  // Bounding box enclosing the route  message BoundingBox  {  required LatLon latDown =1;  required LatLon lonLeft =2;  required LatLon latUp =3;  required LatLon lonRight =4;  }  // Route segment  message Segment  {  required int32 numberOfCoordinates =1;  repeated CoordinateData data =2;  }  // Coordinate container  message CoordinateData  {  required Coordinate coordinate =1;  }  // Container  message Coordinate  {  required LatLon lat =1;  required LatLon lon =2;  }  // Lat lon  message LatLon  {  required fixed32 value =1;  } |

The highest level is a .proto file is Root. It contains three levels of containers: RootContainer, TrackLevel1 and TrackLevel2. At TrackLevel2 we find TrackMetadata (name, bounding box, time), the StartCoordinate and the SegmentData. SegmentData contains one or more Segments, each segment contains the Coordinates.

The total number of Coordinates in all Segments should not exceed 500. If a route to encode contains more than 500 points, skip some intermediate coordinates.

## Workout files (0x00BEnnnn)

Since the newer versions of the software, TomTom added *workouts* for Running and Cycling. Apart from 25 predefined workouts for each activity (which can be slightly adapted by modifying the *intensity level* of the workout), the TomTom MySports site allows for 5 custom workouts for each activity.

A workout consists of a number of *workout steps* the user has to perform to complete the workout. A step defines what the user has to do. There are five possibilities for the ‘length’ defining the extent of the step:

1. Time
2. Distance
3. Reach HR zone
4. Manual

Each step has an ‘intensity’ to be maintained during the step:

1. None – no particular intensity
2. Pace – maintain pace in min/km
3. Speed – maintain speed in km/h
4. Heart Rate zone – maintain a heart rate level

The workouts are defined on the watch in the *overview file* 0x00BEnnnn files. 0x00BE0000 contains the overview of the workouts with the metadata (name, description, intensity level) and a reference to the *workout file*, 0x00BEnnnn, where nnnn≥1, contains the individual workouts defining the steps. Both file types use the Google protobuf format. Next section presents the protobuf definition file:

|  |
| --- |
| ///#################################################################################################  //  // Definition file for the personal workouts (0x00BEnnnn) files  //  //#################################################################################################  syntax = "proto2";  package tutorial;  *<Language specific statements>*  //#################################################################################################  // ROOT level  //#################################################################################################  message Root  {  repeated RootContainer rootContainer =1;  }  //#################################################################################################  // 1st level: root container  //#################################################################################################  message RootContainer  {  optional Metadata metadata =7;  optional DataContainer dataContainer =8;  }  //#################################################################################################  // 2nd level: metadata. Containing just two ints having the same value. Occurs often in one file  //#################################################################################################  message Metadata  {  required fixed32 manufacturer =1; // always 0x1234DAEB  required fixed32 fileType =2; // 0x00080100 - 0x00BE0000, 0x00090100 - 0x00BEnnnn where nnnn>0  }  //#################################################################################################  // 2nd level: data container. Contains various types of data  //#################################################################################################  message DataContainer  {  required SubDataContainer subDataContainer =1;  }  //#################################################################################################  // 3rd level: sub data container  //#################################################################################################  message SubDataContainer  {  repeated Description itemDescription =6; // 0x00BE0000  repeated WorkoutListItem workoutListItem =5; // 0x00BE0000  repeated Description workoutDescription =4; // 0x00BEnnnn  optional Workout workout =7; // 0x00BEnnnn  }  //#################################################################################################  // 4th level: descriptions used in the program  //#################################################################################################  message Description  {  required int32 id =1; // 0=workout name, 1=workout description, >=2 steps  required string description =2; // Name/description of the workout/stage  }  //#################################################################################################  // 4th level: the workout as list of steps  //#################################################################################################  message Workout  {  required int32 name =2; // Name of the workout as refrence to a workoutDescription  optional int32 description =3; // Description of the workout, reference to a workoutDescription  required bytes workoutId =4; // UID identifying the workout file?  optional int32 intensityLevel =6; // Easiest: 19, Easier 9, Standard - ommit, Harder 10, Hardest 20  required int32 Type =10; // 1 Fat burn, 2 Endure, 3 Fitness, 4 Speed, 5 Power, 6 Custom  required int32 unknown11 =11; // value 2  repeated WorkoutStep step =12; // Steps that make up the workout  }  //#################################################################################################  // 4th level: an Item in the Workout list/menu  //#################################################################################################  message WorkoutListItem  {  required fixed32 fileId =1; // File ID representing the workout, like 0x00BE0001  required bytes workoutMd5 =2; // MD5 of the workout file  required int32 itemName =3; // Workout Name in the menu, like '★☆☆30 min'  required int32 itemDescription =4; // Workout Description in the menu, like 'Keep going ...'  required bytes workoutId =5; // 16 bytes, same bytes as workoutId in Program  required int32 intensityLevel =7; // Easiest: 19, Easier 9, Standard 0, Harder 10, Hardest 20  required int32 unknown8 =8; // value=0  required int32 fileSize =9; // Size of the workout file in bytes  required fixed64 activity =10; // 0x0802 for cycling, 0x4181 for running  required int32 type =11; // 1 Fat burn, 2 Endure, 3 Fitness, 4 Speed, 5 Power, 6 Custom  required int32 unknown12 =12; // value=2  required int32 unknown13 =13; // value=1  }  //#################################################################################################  // 5th level: Workout step  //#################################################################################################  message WorkoutStep  {  required WorkoutStepSub stepSub =8;  }  //#################################################################################################  // 6th level: Workout step sub  //#################################################################################################  message WorkoutStepSub  {  required int32 stepName =1; // refers to a description ID (e.g. 'warm up', 'work')  optional int32 stepDescription =2; // refers to a description ID (e.g. 'Get into the ... zone')  required int32 stepNumber =4; // 0, 1, 2, 3,...  required int32 stepType =6; // 1: warm up, 2: work, 3: rest, 4: ?, 5: cool down  optional Length stepLength =7; // Duration or distance (called Length by TomTom)  optional Intensity intensity =8; // Intensity goal (HR zone, pace, speed) of the step; optional  }  //#################################################################################################  // 7th level: Step size in terms of duration or distance  //#################################################################################################  message Length  {  optional int32 time =1; // duration in second  optional int32 distance =2; // distance in mm  optional int32 reachZone =4; // HR zone 1, Easy 2 Fat burn, 3 Cardio, 4 Perform, 5 Peak  optional int32 manual =5; // value is always 1  }  //#################################################################################################  // 7th level: Step intensity in heartrate zone or pace  //#################################################################################################  message Intensity  {  optional int32 heartratezone =1; // @HR Zone: 1 Easy, 2 Fat burn, 3 Cardio, 4 Perform, 5 Peak  optional int32 pace =2; // @pace in milliseconds per km e.g. 312000 = 5:12 min/km  optional int32 speed =3; // @speed in mm/sec  } |

## Races (0x0071xxnn)

Race files define a race for a given activity. The file identifyer is **0x0071xxnn** , where **xx** is the activity code and **nn** the number.

The file format is as shown below. The first 50 bytes is metadata, the remainder of the file defines the lap distances or checkpoints, which .

Integers are little-endian

|  |  |  |  |
| --- | --- | --- | --- |
| Byte | Description | Bytes | Format |
| 0 | Name of the race, like ‘3MI 25M’ | 16 | String |
| 16 | Reserved. 0xff | 16 |  |
| 32 | Unknown | 4 |  |
| 36 | Unknown | 2 |  |
| 38 | Number of checkpoints/laps | 4 | Integer |
| 42 | Duration of the race in seconds | 4 | Integer |
| 46 | Distance of the race in meter | 4 | Integer |
| 50+4\*n | Lap distance in meter. Apparently *the sum* of the 4 bytes is the lap distance (which makes the maximum lap distance 4\*255=1024 meters). | 4 | Integer |
|  |  |  |  |

## Activity types

In several places a code is used to identify the activity. Next table shows the meaning:

|  |  |
| --- | --- |
| Code | Sport |
| 0x00 | Running |
| 0x01 | Cycling |
| 0x02 | Swimming |
| 0x07 | Treadmill |
| 0x08 | Freestyle |
| 0x09 | Gym |
| 0x0a | Hiking |
| 0x0b | Indoor cycling |
| 0x0e | Trail running |
| 0x0f | Skiing |
| 0x10 | Snowboarding |

# USB Interface



## Generic mechanism

The communication to the watch at low level takes place by writing request packets to the write endpoint of the USB device and reading response packets from the read end point.

Request packet (TX):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | N, 0≤N≤252 |
| 0x09 | N+2 | Counter | tx msg type | Payload |

Response packet (RX):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | M, 0≤M≤252 |
| 0x01 | M+2 | Counter | rx msg type | Payload |

Header (green)

1. Start of message  
   0x09 for TX, 0x01 for RX
2. Length of remaining part of the message  
   Including the remaining header bytes
3. Counter  
   Should be increased on each TX. RX reflects the value sent in the corresponding TX. Can be used to check if the response belongs to the request.
4. Message type   
   Usually the msg type in the response is equal to the msg type in the request. The exception is the read data request (MSG\_READ\_FILE\_DATA\_REQUEST)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Device | Vendor ID | Product ID | Read endpoint | Write endpoint | RX/TX Packet size |
| Multisports | 0x1390 | 0x7474 | 0x84 | 0x05 | 64 |
| Spark Music | 0x1390 | 0x7475 | 0x81 | 0x02 | 256 |
| Runner Music | 0x1390 | 0x7475 | 0x81 | 0x02 | 256 |
| Spark Cardio | 0x1390 | 0x7475 | 0x81 | 0x02 | 256 |
| Runner Cardio | 0x1390 | 0x7477 | 0x81 | 0x02 | 256 |
| Adventurer | 0x1390 | 0x7477 | 0x81 | 0x02 | 256 |

In case the packet size is 256 and N<252, the remainder of the packet can be set to 0.

Identified message types:

|  |  |
| --- | --- |
| Message type (cmd) | Description |
| 0x02 | MSG\_OPEN\_FILE\_WRITE |
| 0x03 | MSG\_DELETE\_FILE |
| 0x04 | MSG\_WRITE\_FILE\_DATA |
| 0x05 | MSG\_GET\_FILE\_SIZE |
| 0x06 | MSG\_OPEN\_FILE\_READ |
| 0x07 | MSG\_READ\_FILE\_DATA\_REQUEST |
| 0x09 | MSG\_READ\_FILE\_DATA\_RESPONSE |
| 0x0A | ~~MSG\_FIND\_CLOSE~~ |
| 0x0C | MSG\_CLOSE\_FILE |
| 0x0D | MSG\_UNKNOWN\_0D |
| 0x0E | MSG\_FORMAT\_WATCH |
| 0x10 | MSG\_RESET\_DEVICE |
| 0x11 | MSG\_FIND\_FIRST\_FILE |
| 0x12 | MSG\_FIND\_NEXT\_FILE |
| 0x14 | MSG\_GET\_CURRENT\_TIME |
| 0x1A | MSG\_UNKNOWN\_1A |
| 0x1D | MSG\_RESET\_GPS\_PROCESSOR |
| 0x1F | MSG\_UNKNOWN\_1F |
| 0x20 | MSG\_GET\_PRODUCT\_ID |
| 0x21 | MSG\_GET\_FIRMWARE\_VERSION |
| 0x22 | MSG\_UNKNOWN\_22 |
| 0x23 | MSG\_UNKNOWN\_23 |
| 0x28 | MSG\_GET\_BLE\_VERSION |

## Higher level functions

### Open File/Close File/Delete file

TX:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 4 |
| 0x01 | 6 | cnt | **CMD** | File ID |

RX:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 4 | 4 | 4 | 4 | 4 |
| 0x01 | 22 | cnt | **CMD** |  | File ID |  |  | Error |

|  |  |
| --- | --- |
| Variable | Description |
| CMD | Command byte (message type) |
| File ID | ID of the file. Appears to be 0x00000000 on the adventurer |
| Error | Error indication. 0 – no error, other value - error |

|  |  |
| --- | --- |
| Command (cmd) | Description |
| 0x02 | MSG\_OPEN\_FILE\_WRITE |
| 0x03 | MSG\_DELETE\_FILE |
| 0x06 | MSG\_OPEN\_FILE\_READ |
| 0x0C | MSG\_CLOSE\_FILE |

### Reading files

1. Open file for reading (MSG\_OPEN\_FILE\_READ)
2. Request file size (=bytes to be read; MSG\_GET\_FILE\_SIZE)
3. Repeat: read file data chunk (MSG\_READ\_FILE\_DATA\_REQUEST)
4. Close file (MSG\_CLOSE\_FILE)

Reading file data: File is read in chunks. The amount of bytes to read is defined in the TX message

TX:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 4 | 4 |
| 0x01 | 10 | cnt | **0x07** | File ID | Length |

RX:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 4 | 4 | Read |
| 0x01 | Read+10 | cnt | **0x09** | File ID | Read | File data |

|  |  |
| --- | --- |
| Variable | Description |
| File ID | File to read |
| Length | Bytes to read from the opened file. For the Multisports it is max. 50, for Spark, Runner, Adventurer it is max. 242 bytes. |
| Read | Bytes read. Should be equal to Length |
| File Data | Chuck of file data read. |

### Writing files

1. Open file for writing (MSG\_OPEN\_FILE\_WRITE)
2. Repeat: Write file data chunk (MSG\_WRITE\_FILE\_DATA)
3. Close file (MSG\_CLOSE\_FILE)

Writing file data in chunks:  
File is written in chunks. Therefore the file data has to be split up in chunks. The number of chunks is roundup(fileSize/maxChunkSize). The maxChunkSize depends on the watch type (see below).

TX:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 4 | Length |
| 0x01 | Length+6 | cnt | **0x04** | File ID | Chunk data |

RX:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 4 | 4 | 4 | 4 | 4 |
| 0x01 | 22 | cnt | **0x04** |  | File ID |  |  |  |

|  |  |
| --- | --- |
| Variable | Description |
| File ID | File to write |
| Length | Bytes to write from the opened file. For the Multisports it is max. 54, for other Spark, Runner, Adventurer it is max. 246 bytes. |
| Read | Bytes read. Should be equal to Length |
| File Data | Chuck of file data read. |

### Request file size

This call returns the file size. On the Multisports model the file must be opened for reading first (MSG\_OPEN\_FILE\_READ).

TX:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 4 |
| 0x01 | 6 | cnt | **0x05** | File ID |

RX:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 4 | 4 | 4 | 4 | 4 |
| 0x01 | 22 | cnt | **0x05** |  | File ID |  | Size | Error |

|  |  |
| --- | --- |
| Variable | Description |
| File ID | ID of the file. Appears to be 0x00000000 on the Adventurer |
| Size | File size in bytes |
| Error | >0 when an error occurs: if the file does not exist or the file has not been opened for reading (Multisport model only). 0 If no error occurred. |

### Listing/enumerating files

This method can be used to enumerate files IDs and corresponding lengths.

1. List first file, resets the enumeration (MSG\_FIND\_FIRST\_FILE)
2. Repeat: list subsequent files (MSG\_FIND\_NEXT\_FILE)
3. ~~Close find (MSG\_FIND\_CLOSE)~~

List first file:

TX:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 4 | 4 |
| 0x01 | 10 | cnt | **0x11** | 0 | 0 |

RX:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 4 | 4 | 4 | 4 | 4 |
| 0x01 | 22 | cnt | **0x11** |  | File ID |  | Size | End of list |

List next file:

TX:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 1 |
| 0x01 | 2 | cnt | **0x12** |

RX:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 4 | 4 | 4 | 4 | 4 |
| 0x01 | 22 | cnt | **0x12** |  | File ID |  | Size | End of list |

|  |  |
| --- | --- |
| Variable | Description |
| File ID | ID of the file. |
| Size | File size in bytes |
| End of list | 0 if more files available, otherwise if not. If this value is unequal to 0, File ID and size have no meaning. |

### Get Watch time

TX:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 1 |
| 0x01 | 2 | cnt | **0x14** |

RX:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 4 | 4 | 4 | 4 | 4 |
| 0x01 | 22 | cnt | **0x14** | Time |  |  |  |  |

|  |  |
| --- | --- |
| Variable | Description |
| Time | The current time in epoch seconds (UTC) |

### Firmware version (MSG\_GET\_FIRMWARE\_VERSION)

TX:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 1 |
| 0x01 | 2 | cnt | **0x21** |

RX:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | length |
| 0x01 | 2+length | cnt | **0x21** | Version |

|  |  |
| --- | --- |
| Variable | Description |
| Version | The version as string of *length* bytes, like ‘1.7.53’ |

### Bluetooth Low Energy version (MSG\_GET\_BLE\_VERSION)

TX:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 1 |
| 0x01 | 2 | cnt | **0x28** |

RX:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 4 |
| 0x01 | 6 | cnt | **0x28** | Version |

|  |  |
| --- | --- |
| Variable | Description |
| Version | The version |

### Product ID (MSG\_GET\_PRODUCT\_ID)

TX:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 1 |
| 0x01 | 2 | cnt | **0x20** |

RX:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 4 |
| 0x01 | 6 | cnt | **0x20** | Product ID |

|  |  |
| --- | --- |
| Variable | Description |
| Product ID | Product ID, for Adventurer 0xe0070000. |

### Reset device (MSG\_RESET\_DEVICE)

TX:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 1 |
| 0x01 | 2 | cnt | **0x10** |

The reset is used after uploading firmware files. The reboot installs the firmware. Since the watch resets, no response (RX) is sent.

### Reset GPS processor (MSG\_RESET\_GPS\_PROCESSOR)

TX:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 1 |
| 0x01 | 2 | cnt | **0x1D** |

RX:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | length |
| 0x01 | 2+length | cnt | **0x1D** | Reboot message |

|  |  |
| --- | --- |
| Variable | Description |
| Reboot message | A message as string of *length* bytes, like  ‘wait 1 minute before disconnecting USB’ |

### Format Watch (MSG\_FORMAT\_WATCH)

This method formats the watch. After formatting, it is required to download and write the firmware as described in ‘Firmware files (0x000000f0, 0x00000012, 0x00010200)’. Beware that the required preference file is also deleted during the format, so download this file first before formatting the watch.

Disconnecting the watch after format and before writing the firmware results in the watch asking to be reconnected again:



After the format and writing the firmware the watch can be registered again using TomTom Sports Connect.

TX:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 1 |
| 0x01 | 2 | cnt | **0x0E** |

RX:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 4 | 4 | 4 | 4 | 4 |
| 0x01 | 22 | cnt | **0x0E** |  |  |  |  | Error |

|  |  |
| --- | --- |
| Variable | Description |
| Error | 0 if no error, >0 if an error occurred |