MYLAPS Timing & Scoring TCP-IP V2

Protocol Documentation

VERSION 1.3 (APRIL 2021)

MYLAPS Headquarters
Zuiderhoutlaan 4
2012 PJ Haarlem
The Netherlands
+31 (0)23 760 0100
info@mylaps.com
www.mylaps.com

Contents

Introduction

TCP-IP V2 Protocol was presented in 2012 and since then some adjustments have been made. The previous version of this document had references to some old versions, which made it difficult for external software developers to understand the expected behavior of Timing & Scoring TCP/IP Exporter correctly. This document is written to solve this situation and make one clear reference for all specifications needed.

The current TCP-IP Protocol has two possible versions of which the external server can choose when the connection is established:

- V1, defined in 2009, supported by Timing & Scoring for backwards compatibility with old third-party solutions.
- V2, defined in 2012, that it is the one MYLAPS recommends using

Common specifications

Timing & Scoring TCP/IP Exporter behaves as a socket client. The exporter can connect to a TCP/IP socket server at a chosen port number (default port is 3097).

When the connection is established, the client and server exchange messages with information. These messages are strings and must confirm a protocol.

Message syntax

A message is built up out of the following components:

```
SourceName@Function[@Data]@$
```

Here, `@` is the separator to identify the entities of which the string consists. `SourceName` is the location where the data comes from or, depending on the case, could also be server name. The `Function` part defines which function or service is called. `Data` is optional and can be one string only or consist several strings, each separated by `@`. The `$` identifies the end of the message.

Examples:

```
Bag@GetInfo@$
```

Source name: `Bag`
Function: `GetInfo`

```
Start@Store@DK5M6M9 12:15:05.61 1 1FFFFFFFF 1FFFFFF0908033F@1@$
```

Source name: `Start`
Function: `Store`
Data field 1: `DK5M6M9 12:15:05.61 1 1FFFFFFFF 1FFFFFF0908033F.`
Data field 2: `1`

Version 1

Establishing the connection

The connection between Timing & Scoring (client) and server is initiated by a Pong message that the client sends. The server should return an `AckPong` message with the name in it.

This is the syntax of `Pong` and `AckPong` messages:

```
> (LocationName)@Pong@$
< (ServerName)@AckPong@$
```

Examples:

```
> Finish@Pong@$
< Test@AckPong@$
```

Checking if the connection is open

Both, client and server, have the possibility to send a `Ping` message to the other part to check if the connection is open. When done, the other side should answer with an `AckPing` message.

This is the syntax of `Ping` and `AckPing` messages when the communication is started by the client:

```
> (ClientName)@Ping@$
< (ServerName)@AckPing@$
```

And this is the syntax of `Ping` and `AckPing` messages when the communication is started by the server:

```
< (ServerName)@Ping@$
> (LocationName)@AckPing@$
```

Examples:

```
> T&S@Ping@$
< Test@AckPing@$

< Test@Ping@$
> Start@AckPing@$
> Finish@AckPing@$
```

NOTE: In this case, the client will send an `AckPing` message for each of the active locations

Sending data to the server

The client uses a `Store` message to push passings to the server. This message can contain a maximum of 50 passings. Once the server has received a `Store` message, it should answer the client with an `AckStore` one.

The client will send the same `Store` message every 20 seconds until an `AckStore` message related to that `Store` message is received.

This is the syntax of `Store` and `AckStore` messages:

```
> (LocationName)@Store@(ChipRecord1)[@(ChipRecordN)]@messageNumber@$
< (ServerName)@AckStore@messageNumber@$
```

Where:

- `ChipRecordX` (X could be any number from 1 to 50) is a fixed length string with the following format (known as ReadLite format)
  - 7 characters for the chip code
  - 12 characters for the time of the day (if the precision is less than milliseconds, it should start with blank characters to complete all 12 characters)
  - 2 characters with the device number
  - 2 characters with the reader number
  - 1 character with the hexadecimal number assigned to the antenna (LF) or detections counter clips on 16 (chipX)
  - 3 characters with the lap counter
  - 4 characters with the sequence number (in hexadecimal). This is not used by all devices
  - 6 characters with the date in yyMMdd format
  - 2 characters with a checksum
- `messageNumber` identifies the message and should be a sequential one maintained by the client

Example:

```
> Finish@Store@
0RK9T6N 13:50:23.30 9323 10000090803 87@
0RK9T6N 13:50:25.35 9323 100010908038F@
3@$
< Test@AckStore@3@$
```

Getting device info from the client

The server could, using `GetInfo` message, request some information about the devices that are connected to the client. The client should answer with an `AckGetInfo` message containing all devices connected to Timing and Scoring and the location where they are assigned.

This is the syntax of `GetInfo` and `AckGetInfo` messages:

```
< (ServerName)@GetInfo@$
> (LocationName)@AckGetInfo@[(DeviceName)@Unknown@(ComputerName)]$
```

Where:

- `DeviceName` is the name of the device in Timing & Scoring
- `ComputerName` is the name of the computer where Timing & Scoring is running (this name comes from the Windows settings)

Examples:

```
< Test@GetInfo@$
> Finish@AckGetInfo@PalicanEar11@Unknown@joost_sys_xp$
Finish@AckGetInfo@PalicanEar12@Unknown@joost_sys_xp$
10 KM@AckGetInfo@Apex_3@Unknown@joost_sys_xp$
```

Version 2

Introduction

TCP-IP V2 Protocol is designed as an extension to version 1 protocol. Reasons for this extension is:

- Flexibility. By specifying data in key/value pairs, the protocol is not as rigid as the previous protocol, which used the fixed width ReadLite line format. This flexibility leads to fewer limitations on the data that is sent.

  In the protocol version 2, the server can specify which data it is interested in to receive. A server can request the minimum data needed for timing purposes in order to keep the message size small or can request more detailed data on passings and device information to use for diagnostics and monitoring purposes.

- Performance. The performance of the version 1 protocol was constrained by the round-trip time to the server Timing & Scoring was communicating with. As the messages contained a maximum of 30 passings, and Timing & Scoring would wait for an acknowledgement before sending the next message, the communication was limited by network latency and not throughput. This performance hit was much more noticeable when sending times over a WAN connection with higher latencies compared to those experiences on a LAN connection.

  The version 2 protocol sends a maximum of 100 passings per message and allows up to 10 messages to be sent and unacknowledged before a 'wait for acknowledgement' is required. In internal testing, this has resulted in a performance increase of more than 100 times over a WAN connection.

- Future improvements. By formatting messages as key/value pairs and not a fixed width format, it will be easier to extend the protocol in the future to accommodate new data requirements. New keys can be added to the protocol without effecting the existing communication and older servers, which do not recognize the new keys, can ignore them or report them to the administrator.

The version 2 protocol is not intended to replace the version 1 but is designed as an extension to it.

Note that the order of messages is not guaranteed. As Timing & Scoring no longer sends a single message and waits for its acknowledgement, it is possible to receive messages out of order. It is up to the server implementer to handle this situation; Timing & Scoring does not guarantee that you will receive messages in the order of their acknowledgement number. The order of key/value pairs within a message is also not guaranteed.

Establishing the connection

The connection between Timing & Scoring (client) and server is initiated by a `Pong` message that the client sends. The server should return an `AckPong` message with the name in it.

This is the syntax of `Pong` and `AckPong` messages:

```
> (LocationName)@Pong@$
< (ServerName)@AckPong@Version2.1@$
```

Examples:

```
> Finish@Pong@$
< Test@AckPong@Version2.1@$
```

Checking if the connection is open

Both, client and server, have the possibility to send a `Ping` message to the other part to check if the connection is open. When done, the other side should answer with an `AckPing` message.

This is the syntax of `Ping` and `AckPing` messages when the communication is started by the client:

```
> (ClientName)@Ping@$
< (ServerName)@AckPing@$
```

And this is the syntax of `Ping` and `AckPing` messages when the communication is started by the server:

```
< (ServerName)@Ping@$
> (ClientName)@AckPing@$
```

Examples:

```
> T&S@Ping@$
< Test@AckPing@$

< Test@Ping@$
> Start@AckPing@$
> Finish@AckPing@$
```

NOTE: In this case, the client will send an `AckPing` message for each of the active locations

Sending data to the server

Client uses `Passing` message to push passings to the server. This message can contain a maximum of 100 passings. Once the server has received a `Passing` message, it should answer the client with an `AckPassing` one.

The client will send the same `Passing` message every 20 seconds until an `AckPassing` message related to that `Passing` message is received.

This is the syntax of `Passing` and `AckPassing` messages:

```
> (LocationName)@ Passing@(ChipRecord1)
[@(ChipRecordN)]@messageNumber@$
< (ServerName)@AckPassing@messageNumber@$
```

Where `ChipRecordX` is a structure with following syntax:
`passingXkey1=passingXvalue1|...|passingXkeyN=passingXvalueN`

Example:
```
> 10KM@Passing@
t=13:11:30.904|c=0000041|ct=UH|d=120606|l=13|dv=4|re=0|an=00001111|g
=0|b=41|n=41@
t=13:12:21.830|c=0000039|ct=UH|d=120606|l=30|dv=4|re=0|an=00001101|g
=0|b=39|n=39@1016@$
< Test@AckPassing@1016@$
```

Sending markers to the server

The client uses `Marker` message to push markers to the server. This message can contain a maximum of 100 markers. Once the server has received a `Marker` message, it should answer the client with an `AckMarker` one.

The client will send the same `Marker` message every 20 seconds until an `AckMarker` message related to that `Marker` message is received.

This is the syntax of `Marker` and `AckMarker` messages:
```
> (LocationName)@Marker@(MarkerRecord1)
[@(MarkerRecordN)]@messageNumber@$
< (ServerName)@AckMarker@messageNumber@$
```

Where `MarkerRecordX` is a structure with following syntax:
`markerXkey1=markerXvalue1|...|markerXkeyN=markerXvalueN`

Example:
```
> Finish@Marker@t=11:03:40.347|mt=Gunshot|n=Gunshot 1@4@$
< Test@AckMarker@4@$
```

Getting device info from the client

The server could, by using a `GetInfo` message, request some information about the devices connected to a specific location. The client should answer with an `AckGetInfo` message containing all devices connected to Timing & Scoring and the location where they are assigned.

This is the syntax of `GetInfo` and `AckGetInfo` messages:
```
< (ServerName)@GetInfo@(LocationName)@$
> (LocationName)@AckGetInfo@(DeviceInfoRecord1)@$
  [(LocationName)@AckGetInfo@(DeviceInfoRecordN)@$
```

Where `DeviceInfoRecordX` is a structure with following syntax:
`deviceInfoXkey1=deviceInfoXvalue1|...|deviceInfoXkeyN=deviceInfoXval ueN`

Example:

```
< Test@GetInfo@20M@$
> 20M@AckGetInfo@
id=20250558687|n=BibTagDecoder00DF|mac=0004B70700DF|ant=2|time=95446
3123529@$
20M@AckGetInfo@
id=20250558568|n=BibTagDecoder00AA|mac=0004B70700AA|ant=1|time=95446
3123529@$
```

Getting race info from the client

The server could, using `GetInfo` message, request some information about the race. The client should answer with an `AckGetInfo` message.

This is the syntax of `GetInfo` and `AckGetInfo` messages:

```
< (ServerName)@GetInfo@$
> T&S@AckGetInfo@(RaceInfoRecord)@$
```

Where `RaceInfoRecord` is a structure with following syntax:

```
raceInfoXkey1=raceInfoXvalue1|...|raceInfoXkeyN=raceInfoXvalueN
```

Examples:

```
< Test@GetInfo@$
> T&S@AckGetInfo@sd=2012-06-21|rn=My 5K Race@$
```

Getting location info from the client

It is possible for the server, using `GetLocations` message, to request a list of active locations in Timing & Scoring. The client answer with `GetLocations` message

This is the syntax of `GetLocations` message:

```
< (ServerName)@GetLocations@$
> Toolkit@GetLocations@(LocationInfoRecord)@$
```

Where `LocationInfoRecord` is a structure with following syntax:

```
ln=(LocationName1)@...@ln=(LocationNameN)
```

Examples:

```
< Test@GetLocations@$
> Toolkit@GetLocations@ln=Start@ln=5K@ln=Finish@$
```

Specify parameters to report

By default, version 2 will send the following data:

- Passing
  - Chip code
  - Chip type
  - Time
  - Date
  - Lap
  - Device number
  - Reader number
  - Antenna number
  - Group id
  - Bib text
  - Bib
- Device
  - Name
  - Number
  - MAC
  - Antenna count
  - Time

The server can specify a custom set of data to be received from Timing&Scoring;

To specify a custom set of data for passings, the server responds to a Pong request with the following format:

```
> (LocationName)@Pong@$
< (ServerName)@AckPong@Version2.1@param1[|paramN]@$
```

Example:

```
> 20K@Pong@$
< Test@AckPong@Version2.1@c|ct@$
```

To specify a custom set of data for passings and device info, the server responds to a Pong request with the following format:

```
> (LocationName)@Pong@$
<
(ServerName)@AckPong@Version2.1@param1[|paramN]@deviceParam1[|device
ParamN]@$
```

Example:

```
> 20K@Pong@$
< Test@AckPong@Version2.1@c|ct@id|n|dt@$
```

Possible `paramX` and `deviceParamX` values can be found in the appendix to this document.

Appendix: Version 2 Parameters

Race Info Parameters

Field	Identifier	Description	Example Values
Race name	rn	Name of the race	*Valley Park Triathlon* *Run Fast 5K*
Start date	sd	Start date of the race in YYYY-MM-DD format	*2010-05-22* *2012-06-12*

Marker Parameters

Field	Identifier	Description	Example Values
Type	mt	What type of marker this is	*Gunshot* *Marker* *ext-1*
Time	t	Time of the day where this marker occurs in hh:mm:ss.001 format	*11:03:40.347* *15:12:12.333*
Name	n	Name of the marker	*Gunshot 1* *Marathon Start*

Passing Parameters

Field	Identifier	Description	Example Values
Chip code	c	7-character chip code	*DX93845* *0000345* *4AYX3TF*
Chip type	ct	The type of chip this is	*AP* *LF* *CX* *UH* *NOTCX* *UNKNOWN*
Date	d	Date of passing in YYMMDD format	*120122* *101210*
Lap number	l	Lap count	*0* *1* *2*
Device number	dv	Number of the device where this passing has been recorded	*0* *1* *2*
Reader number	re	Number of the reader where this passing has been recorded	*1* *8*
Antenna number	an	Championship: Number of the antenna where this passing has been recorded	*1* *2* *5*
		BibTag: Antenna mask	*00010000* *11000000*
Group id	g	Group id of this passing (BibTag)	*0* *2*
Bib number	b	Bib number associated to the passing (BibTag)	*1* *1001*
Bib text	n	Bib text associated to the passing (BibTag)	*F-10145* *M123* *554*
Time	t	Time of the day where this passing occurs in hh:mm:ss.001 format	*14:22:33.091* *07:00:00.550*

Field	Identifier	Description	Example Values
UNIX Time	ut	Time of the day of this passing in milliseconds (UNIX format)	*18274599* *192837288*
UTC Time	utc	Time of the day of this passing in milliseconds on UTC (UNIX format)	*11273599* *232827287*
Hit count	h	Number of detections (ProChip/BibTag)	*10* *65*
Time source	ts	Time source the device was using when this passing was recorded	*Unknown* *Manual* *GPS* *NTP*
Batch id	bid	Batch id of this passing (BibTag)	*23-33290* *12-34123*
Amplitude	am	Amplitude in points (ProChip/BibTag)	*15* *125*
Amplitude (in dBm)	amd	Amplitude in dBm (ProChip/BibTag)	*-101.15* *-66.15*
MAC Address	dm	MAC address of the device this passing has been recorded	*no-device* *no-mac* *0030568F2CE*
Strongest Antenna	ans	Strongest antenna id (BibTag)	*1* *5*
Average Antenna	ana	Average antenna (BibTag)	*1.7* *2.3*

Device Parameters

Field	Identifier	Description	Example Values
Device id	id	Unique id assigned to the device by Timing&Scoring	*1* *3*
Device name	n	Name of the device	*BibTag-0345,* *Finish Backup*
Device type	dt	Type of device	*ChampionChip Portable Decoder* *ChampionChip Chip Scanner* *ChampionChip Ear 1* *ChampionChip Ear 2* *ChampionChip Portable Controller* *ProChip Portable Decoder* *ProChip Smart Decoder* *Timepoint System Decoder* *ProChip Decoder* *BibTag Decoder* *Import Device* *HandReader* *Unknown device*
Device number	nr	Number assigned to the device by the timer	*1* *5*
Device MAC	mac	MAC address of the device	*no-mac* *0030568F2CE*

Field	Identifier	Description	Example Values
Battery level	bat	Value (integer) which indicates the battery level. 0 means dead battery and -1 unknown power level	*-1* *0* *50*
Time Between Same Chip	tbsc	Time between same chip, in seconds (ChampionChip) or milliseconds (BibTag). -1 means unknown	*0* *10* *1000* *-1*
Profile	prof	Device profile (BibTag)	*Unknown* *Main* *Backup* *Scanner* *Custom*
Antenna Count	ant	Number of antennas connected to the device (BibTag)	*0* *8*
Firmware Version	fwv	Current firmware version on the device	*n.a.* *4.7* *5.0f*
Beeper Volume	bvol	Beeper volume level, from 0 to 3 (BibTag/ProChip)	*0* *3* *n.a.*
Beep Type	btyp	Beep type. 0 means continuous and 1 means single (BibTag)	*0* *1* *n.a.*
Continuous Mode	cont	Indicates if Continuous Mode is enabled. Possible values are True and False (BibTag)	*true* *false* *n.a.*
Gun Holdoff	gho	Gun Holdoff in milliseconds (BibTag/ProChip)	*0* *50* *n.a.*
Ext1 Holdoff	ex1ho	Ext1 Holdoff in milliseconds (BibTag/ProChip)	*0* *50* *n.a.*
Ext2 Holdoff	ex2ho	Ext2 Holdoff in milliseconds (BibTag/ProChip)	*0* *50* *n.a.*
Temperature	temp	Device temperature in Celsius (BibTag/ProChip)	*0* *30* *n.a.*
Daylight Savings Time	dst	Indicates if daylight saving time is enabled. Possible values are True and False (BibTag/ProChip)	*true* *false* *n.a.*
GPS Satellite Count	gpsc	Number of GPS satellites connected to (BibTag/ProChip)	*0* *6* *n.a.*
GPS Longitude	gpsx	Longitude (x) value reported by GPS (BibTag/ProChip)	*12.0432* *n.a.*
GPS Latitude	gpsy	Latitude (y) value reported by GPS (BibTag/ProChip)	*30.1123* *n.a.*
Timezone	tz	Time zone offset from GMT in minutes, from -720 to 720 (BibTag/ProChip)	*-720* *300* *720* *n.a.*

Field	Identifier	Description	Example Values
Time source	ts	Device time source	*Unknown* *Manual* *GPS* *NTP*
Power Source	ps	Device power source	*net* *internal* *external* *charging*
Reader Channels	rcs	Reader channels of the decoder (BibTag)	*A* *CD* *n.a.*
GPRS APN	gprsa	APN of mobile communication (BibTag/ProChip)	*wap.verizon* *n.a.*
IP Address	ip	Device's IP Address	*72.12.123.1* *n.a.*
Connection Type	con	Device connection type	*TCP/IP* *Serial* *Modem* *CCNet* *None*
CCNetServer 1 Address	ccn1	Device CCNetServer URL/IP for first option	*server1.championchip.net* *usa1.championchip.net*
CCNetServer 2 Address	ccn2	Device CCNetServer URL/IP for second option	*server1.championchip.net* *usa1.championchip.net*
CCNetServer Connection Method	ccnc	Device connection method to CCNetServer	*Unknown* *Ethernet* *GSM* *Off*
Time	time	Current device time in milliseconds ***Warning: this report of time should not be used for any timing purposes! It's accuracy is not guaranteed!***	*1292929* *33401929030*
Device is connected	isconn	Is the device connected?	*true* *false*

Location Parameters

Field	Identifier	Description	Example Values
Location name	Ln	Name of the location as it is written in Timing & Scoring	*Start* *Finish*

Support

In case you encounter any issues, please contact your sales office:

MYLAPS EMEA Office Haarlem
Haarlem
The Netherlands
Tel: +31 23 7600100
E-mail: info@mylaps.com

MYLAPS Americas Office
Atlanta
USA
Tel: +1 (678) 816 4000
E-mail: info.americas@mylaps.com

MYLAPS Japan Office
Tokyo
Japan
Tel: +81 3 6418 8209
Email: info.japan@mylaps.com

MYLAPS Asia Pacific Office
Sydney
Australia
Tel: +61 (0)2 7201 8435
Email: info.asia.pacific@mylaps.com

MYLAPS Asia Office
Selangor
Malaysia
Tel: +60 (0)356131235
Email: info.asia@mylaps.com

