QTM Real-time Server Protocol Documentation

Version 1.22

Introduction.

The Qualisys Track Manager software is used to collect and process motion capture data from Qualisys motion capture cameras. The software is running under Windows and offers both post-processing and real-time processing functionality. The processed real-time data can be retrieved from QTM over a TCP/IP (or UDP/IP) connection in real-time. This document describes the protocol used in such a connection.

Protocol version

This document describes version 1.22 of the QTM RT server protocol.

Standard

QTM is backwards compatible with all previous versions of the protocol. The QTM RT server keeps track of the protocol version used by each RT client connected to it, and adapts the data to be sent to each client according to their selected protocol version.

To ensure that a particular client will work with all future releases of QTM, the client only needs to send the `Version` command to the QTM RT server when connecting to it.

At the end of this document there is a list of the changes that have been made to the protocol between different versions.

Open sound control

Version 1.6 and later of the QTM RT server protocol supports the *OSC (Open Sound Control)* protocol over UDP. Connecting to the RT server when using OSC, differs from the standard version of the RT protocol. See Connecting.

Overview.

Protocol details

Standards used

The QTM RT server should be able to communicate successfully with clients from any computer architecture. To avoid problems, check the points below.

Byte order

The byte order of data pieces larger than one byte can differ between computer architectures. Select the byte-order your computer architecture prefers by connecting to the corresponding TCP/IP port on the QTM RT server. See IP port numbers.

Floating point values

The floating point type used by the QTM RT server is the standard defined by IEEE 754. Single precision floats (32-bit) values are used.

Auto discover

It is possible to auto discover any computers running QTM version 2.4 (build 551) or later on your local area network. This is done by broadcasting an UDP packet to the QTM auto discover port, see IP port numbers. The discover packet shall contain the port number to which QTM sends an UDP response string, see Discover packet. Except for the IP address, the client will also respond with the host name, QTM version and number of connected cameras.

Connecting

Connecting to the QTM RT server is simply a matter of connecting to a specific TCP/IP port on the computer where QTM is running.

The first thing that happens when you have connected to the QTM RT server is that the server sends a welcome message string:

```
QTM RT Interface connected.
```

Number of simultaneous connections is limited to 10. If the limit is reached while connecting, QTM will respond with an error message:

```
Connection refused. Max number of clients reached.
```

The first command that the client should send to the server is the `Version` command, to make sure that QTM is using the RT protocol version expected by the client. If the client doesn't send the `Version` command, QTM will use version 1.1.

If the client will request streaming data over TCP/IP (default) or polled data, make sure to disable **Nagle's algorithm** for the TCP/IP port. See Disabling Nagle's algorithm.

Disabling Nagle's algorithm

The TCP protocol by default uses a performance improvement called Nagle's algorithm that reduces the bandwidth used by the TCP connection. In the case of a real-time server that sends small amounts of data in each frame, this algorithm should be turned off. Otherwise the server (and client) will wait to fill a full TCP packet, or until the previous packet has been acknowledged by the receiver, before sending it to the client (or the server).

On the Windows platform, Nagle's algorithm can be turned off by enabling the **TCP_NODELAY** option for the TCP/IP port.

If you use UDP/IP streaming only (via the `StreamFrames` command), it is *not* necessary to turn off Nagle's algorithm for the TCP/IP port, since a little higher latency can be accepted in the parts of the protocol that do not stream data in real-time. The UDP streaming protocol has no such bandwidth optimization and is designed for low latency-applications.

IP port numbers

In the **RT output** tab of the Workspace Options dialog in QTM, you can configure the QTM RT server ports.

You can only edit the base port (**22222** by default). This is is the legacy server port, for version 1.0 of the protocol. All other ports except for the auto discover port are set from the base port. See table below.

PORT	DEFAULT	DESCRIPTION
Base port-1	22221	Telnet port. Used mainly for testing. Connects to the latest version of the RT protocol.
Base port	22222	Supports only the 1.0 version of the protocol. **Don't use this port for any new clients.**
Base port+1	22223	Little-endian version of the protocol. Used from protocol version 1.1 and onwards.
Base port+2	22224	Big-endian version of the protocol. Used from protocol version 1.1 and onwards.
Base port+3	22225	QTM RT-protocol over OSC (Open Sound Control) protocol. OSC protocol is sent over UDP.
22226	22226	QTM auto discover. QTM listens for UDP discover broadcasts on this port and responds with an UDP message to the sender.

Protocol structure

All data sent between the server and the client is packaged in packets with an **8-byte** header consisting of a **4-byte Size** field and a **4-byte Type** field.

In most cases, the QTM RT server does not send any data to the client unless requested. The client sends a command and the QTM RT server sends a response in form of a string or XML data or frame data. The client should however be able to handle cases when packets arrive which is not a response to a command. For example, an event or an error message could arrive when a completely different response is expected.

Reading settings

Before requesting streamed data, it may be necessary to ask QTM about different settings, for example capture frequency and marker labels. For all such information that does not change with each frame, the command `GetParameters` is used.

If `GetParameters` succeeded, the server will send an XML packet, with the requested information.

Otherwise an error packet will be sent:

```
Parse error
```

See XML packet, for more details on which settings that are available.

Change settings

It is possible to change some of the QTM settings via the RT server. This is done by sending an XML packet, containing the settings to be changed. Settings that are possible to change are: General, 6d, Image, Force and skeleton.

If the settings were updated ok, the server will send a command packet in response:

```
Setting parameters succeeded
```

Otherwise a error packet will be sent:

```
Setting parameters failed
```

Change settings is not available with the OSC protocol.

Streaming data

The client has two options when requesting data frames from the QTM RT server: polling mode or streaming mode.

In polling mode, the client requests each frame in the pace it needs them, using the command *GetCurrentFrame*.

In streaming mode, the client tells QTM to stream data at a fixed rate to the client by using the *StreamFrames* command. QTM keeps streaming data until the measurement is stopped in QTM or the client tells QTM to stop.

In either mode, the client decides what type of data it needs (2D, 3D, 6D, Analog, Force or a combination of these).

In streaming mode, the client may request streaming over UDP/IP instead of TCP/IP, to minimize the protocol latency (at the cost of possibly losing some data frames). When using the OSC protocol, all data is sent via UDP.

Commands.

In the description of the commands, number parameters are designated by an `n`, optional parameters are designated by enclosing brackets `[ ]` and choices between possible values are designated by a `|`. Parentheses are used to group parameters together. None of these characters, ie brackets `[ ]`, the pipe character `|` or parentheses `()` should be included in the command sent to the server.

Command strings and their parameters never contain spaces, so a space character (ASCII 32) is used as separator between command names and parameters.

Command strings and parameter strings are case insensitive.

The response to a command is a command packet, error packet, XML packet, C3D data packet or QTM data packet. Each command below has an example. The examples list all available responses for each command. Command strings and error strings are shown in italic. If the command is not recognized by the server, it will send an error response with the string 'Parse error'.

Table of all commands in the QTM rt server. Parameters in *italics* are variables.

COMMAND	PARAMETERS
Version	[*n*.*n*]
QTMVersion	
ByteOrder	
GetState	
GetParameters	All | ([General] [Calibration] [3D] [6D] [Analog] [Force] [Image] [GazeVector] [EyeTracker] [Skeleton])
GetCurrentFrame	[2D] [2DLin] [3D] [3DRes] [3DNoLabels] [3DNoLabelsRes] [Analog[:*channels*]] [AnalogSingle[:*channels*]] [Force] [ForceSingle] [6D] [6DRes] [6DEuler] [6DEulerRes] [Image] [GazeVector] [EyeTracker] [Timecode] [Skeleton[:global]]
StreamFrames	Stop | ((FrequencyDivisor:*n* | Frequency:*n* | AllFrames) [UDP[:*address*]:*port*] ([2D] [2DLin] [3D] [3DRes] [3DNoLabels]

	[3DNoLabelsRes] [Analog[:*channels*]] [AnalogSingle[:*channels*]] [Force] [ForceSingle] [6D] [6DRes] [6DEuler] [6DEulerRes] [Image] [GazeVector] [EyeTracker] [Timecode] [Skeleton[:global]]))
TakeControl	[*password*]
ReleaseControl	
New	
Close	
Start	[RTFromFile]
Stop	
Load	*filename*
Save	*filename* [Overwrite]
LoadProject	*project_path*
GetCaptureC3D	
GetCaptureQTM	
Trig	
SetQTMEvent	*label*
Reprocess	
Calibrate	[Refine]
Led	*camera* (On | Off | Pulsing) (Green | Amber | All)
Quit	

Version

> **`Version`** `[n.n]`

The first thing that a client should do after connecting to the QTM RT server is to send the Version `command` to the server with the desired protocol version. This will ensure that the protocol described in this document is followed by the server. The server will respond with Version set to n.n, where n.n is the version selected. If no argument is used, the server will respond with the current version.

If you don't set the protocol version yourself, QTM will set it to **version 1.1** by default.

Response

The command will return a command packet or an error packet.

```
Response: 'Version set to 1.22'

Errors:   'Cannot change version while streaming data'
          'Version 1.0 is not supported on this RT client p
ort'
          'Only version 1.0 is supported on this RT client
 port'
          'Version NOT supported'
          'Parse error'
```

QTMVersion

> `QTMVersion`

Returns the QTM version on which the RT server is running.

Response

The command will return a command packet or an error packet.

```
Response: 'QTM Version is 2.3 (build 464)'

Errors:   'Parse error'
```

ByteOrder

> `ByteOrder`

Returns the current byte order.

Response

The command will return a command packet or an error packet.

```
Response: 'Byte order is little endian'
          'Byte order is big endian'

Errors:   'Byte order can only be changed by connecting to
 a different port'
          'Parse error'
```

GetState

```
GetState
```

This command makes the RT server send current QTM state as an event data packet. The event packet will only be sent to the client that sent the GetState command. If the client is connected via Telnet, then the response will be sent as an ASCII string. `GetState` will not show the **Camera Settings Changed**, **QTM Shutting Down** and **Capture Saved events**.

Response

The command will return an event packet or an error packet.

```
Error: 'Parse error'
```

GetParameters

```
GetParameters All | ([General] [Calibration] [3D]
    [6D] [Analog] [Force] [Image] [GazeVector]
    [EyeTracker] [Skeleton[:global]])
```

This command retrieves the settings for the requested component(s) of QTM in XML format. The XML parameters are described here.

By default, skeleton data is in local coordinates. The Skeleton:global argument will change the skeleton data to global coordinates.

Response

The command will return an XML packet or an error packet.

```
Error: 'Parse error'
```

GetCurrentFrame

```
GetCurrentFrame [2D] [2DLin] [3D] [3DRes]
    [3DNoLabels] [3DNoLabelsRes] [Analog[:channels]]
    [AnalogSingle[:channels]] [Force] [ForceSingle]
    [6D] [6DRes] [6DEuler] [6DEulerRes] [Image]
    [GazeVector] [EyeTracker] [Timecode]
    [Skeleton[:global]]
```

The optional channels for Analog and AnalogSingle, is a string containing a list of channels to read from the server. The channels are separated by a `,` and can also contain ranges defined by a `-`. Here is an example: `1,2,3-6,16`

By default, skeleton data is in local coordinates. Skeleton:global will change the skeleton data to global coordinates.

This command returns the current frame of real-time data from the server.

Points worth noting are:

- The frame is composed of the parts specified in the parameters to the command. The exact layout of the data frame in different situations is described in Data packet.
- The composition of the data frame may vary between frames. This is due to the fact that some data (Analog and Force data) is not collected or buffered at the same rate as the camera data (**2D**, **3D**, **6D**). If you specify Analog or Force data to be streamed together with some form(s) of camera data, some data frames may include analog while others don't include it. This is because QTM sends the Analog and Force data as soon as it is available, and it is usually available in fairly large chunks and not as often as camera data is available.
- If there is no ongoing measurement (either it has not started or it has already finished), an empty data frame is sent to the client .
- If a measurement is ongoing but there is no new frame of data available, the server waits until the next frame of data is available before sending it to the client.

Response

The command will return a data packet or an error packet.

```
Error: 'Cannot send current frame while streaming data'
       'Version not supported on current port'
       'Parse error'
```

StreamFrames

```
StreamFrames Stop | ((FrequencyDivisor:n |
    Frequency:n | AllFrames) [UDP[:address]:port] [2D]
    [2DLin] [3D] [3DRes] [3DNoLabels] [3DNoLabelsRes]
    [Analog[:channels]] [AnalogSingle[:channels]]
    [Force] [ForceSingle] [6D] [6DRes] [6DEuler]
    [6DEulerRes] [Image] [GazeVector] [EyeTracker]
    [Timecode] [Skeleton[:global]])
```

The optional channels for Analog and AnalogSingle, is a string containing a list of channels to read from the server. The channels are separated by a `,` and can also

```
contain ranges defined by a `-`. Here is an example: `1,2,3-6,16`
```

By default, skeleton data is in local coordinates. Skeleton:global will change the skeleton data to global coordinates.

This command makes the QTM RT server start streaming data frames in real-time.

Points worth noting are:

- Each frame is composed of the parts specified in the parameters to the command. The exact layout of the data frame in different situations is described in Data packet.

  The composition of the data frame may vary between frames. This is due to the fact that some data (Analog and Force data) is not collected or buffered at the same rate as the camera data (2D, 3D, 6D). If you specify Analog or Force data to be streamed together with some form(s) of camera data, some data frames may include analog while others don't include it. This is because QTM sends the Analog and Force data as soon as it is available, and it is usually available in fairly large chunks and not as often as camera data is available.

- If there is no ongoing measurement (either it has not started or it has already finished), an empty data frame is sent to the client.
- The actual rate at which the frames are sent depends on several factors. Not just the frequency specified in the command parameters:
- The measurement frequency used when acquiring the camera data (2D, 3D, 6D). The transmission rate cannot be greater than this frequency.
  - The real-time processing frequency set in QTM. This may differ greatly from the measurement frequency. For example QTM may be measuring at 1000 Hz but trying to calculate real-time frames only at 50Hz. The transmission rate cannot be greater than this frequency either.
  - The processing time needed for each frame of data in QTM. This may also be a limiting factor – QTM may not have time to process and transmit frames at the rate specified as the real-time processing frequency.
  - The frequency specified by the client in the command parameters. The client has three ways of specifying the preferred data rate of the server. If the client specifies a higher rate than it can receive and handle in real-time, buffering will occur in the TCP/IP or UDP/IP stack at the client side and the client will experience lagging.
  - FrequencyDivisor:n With this setting, QTM transmits every n:th processed real-time frame to the client. Please note that this may not be the same as every n:th frame of the measurement (see real-time processing frequency above).

    Example: QTM is measuring in 200 Hz and real-time tracking in 100 Hz. If a client specifies FrequencyDivisor:4 QTM will send data at a rate of 25Hz.

  - Frequency:n With a specific frequency setting, the QTM RT server will transmit frames at the nearest multiple of the real-time processing frequency.

Example: QTM is measuring in 200 Hz and real-time tracking in 100 Hz. If a client specifies Frequency:60 QTM will send data at an approximate rate of 50Hz.

```
* AllFrames
When a client specifies AllFrames in the StreamFrames comman
d, every real-time frame processed by QTM is transmitted to t
he client.
```

- UDP notes:
  - If the UDP argument is present, the server will send the data frames over UDP/IP instead of TCP/IP. With high network load the risk of losing packets increases. When using TCP/IP, these packets will be retransmitted and no packets will be lost, but on the other hand, when packets are lost the client will not receive any data until they have been retransmitted, which can take up to a second in some cases.

    When using UDP/IP, lost packets are lost, but the next transmitted packet will not be delayed by waiting for retransmissions, so the latency can be a lot better using UDP/IP.
  - The address parameter is optional. If omitted, the UDP frames will be sent to the IP address that the command is sent from (the IP address of the client).
  - The port parameter is not optional. Valid port numbers are 1023 – 65535.
  - When using UDP one cannot be sure that all components are sent in a single data frame packet. It can be divided into several data frame packets. The server will try to fit as many components into one UDP datagram as possible.
- When the measurement is finished, or has not yet started, a special empty data frame packet signaling that no data is available is sent to the client.
- To stop the data stream before it has reached the end of the measurement or to prevent data from being sent if a new measurement is started after the first was finished: send the StreamFrames Stop command.

Response

The command will start streaming data packets or an error packet will be sent.

```
Error: 'Version not supported on current port'
       'Parse error'
```

TakeControl

```
TakeControl  [Password]
```

Password:

The password argument is optional and is only needed if it is required by QTM. QTM can be configured to deny all clients control, only allow clients with correct password or allow all clients control.

This command is used to take control over the QTM RT interface. Only one client can have the control at a time. Once a user has the control, it is possible to change settings, create a new measurement, close measurement, start capture, stop capture and get a capture.

Response

The command will return a command packet or an error packet.

```
Response: 'You are now master'
          'You are already master'

Error:    '192.168.1.5 (1832) is already master'
          'Client control disabled in QTM'
          'Wrong or missing password'
          'Parse error'
```

ReleaseControl

> **`ReleaseControl`**

Release the control over the QTM RT interface, so that another client can take over the control.

Response

The command will return a command packet or an error packet.

```
Response: 'You are now a regular client'
          'You are already a regular client'

Error:    'Parse error'
```

New

> **`New`**

This command will create a new measurement in QTM, connect to the cameras and enter RT (preview) mode. It is only possible to issue this command if you have the control over the QTM RT interface. See TakeControl.

Response

The command will return a command packet or an error packet.

```
Response: 'Creating new connection'

Errors:   'Already connected'
          'Parse error'
          'The previous measurement has not been saved or c
losed'
```

```
           'Parse error'
           'You must be master to issue this command'
```

Close

```
Close
```

This command will close the current QTM measurement. If in RT (preview) mode, it will disconnect from the cameras end exit RT (preview) mode. Otherwise it will close any open QTM measurement file. If the measurement isn't saved, all data will be lost. If QTM is running RT from file, the playback will stop and the file will be closed. It is only possible to issue this command if you have the control over the QTM RT interface. See TakeControl.

Response

The command will return a command packet or an error packet.

```
Responses: 'Closing connection'
           'Closing file'

Errors:    'No connection to close'
           'Parse error'
           'You must be master to issue this command'
```

Start

```
Start  [RTFromFile]
```

This command will start a new capture. If the argument RTFromFile is used, QTM will start streaming real-time data from current QTM file. If there is any file open. It is only possible to issue this command if you have the control over the QTM RT interface. See TakeControl.

Response

The command will return a command packet or an error packet.

```
Responses: 'Starting measurement'
           'Starting RT from file'

Errors:    'Measurement is already running'
           'Not connected. Create connection with new'
           'RT from file already running'
           'No file open'
           'Parse error'
           'You must be master to issue this command'
```

Stop

```
Stop
```

This command will stop an ongoing capture or playback of RT from file. It is only possible to issue this command if you have the control over the QTM RT interface. See TakeControl.

Response

The command will return a command packet or an error packet.

```
Response: 'Stopping measurement'

Errors:   'No measurement is running'
          'Parse error'
          'You must be master to issue this command'
```

Load

```
Load  Filename
```

Filename: A string containing the name of the QTM file to load. If the filename doesn't end with ".qtm", it will be added to the end of the filename. The file name can be a relative or absolute path. See below.

This command will load a measurement from file. The name of the file is given in the argument. The file name can be relative or absolute. If the file name is relative, QTM will try to find the file in the data folder located in the project folder. If the file doesn't exist, current measurement isn't saved or an active camera connection exists, the measurement will not load.

It is only possible to issue this command if you have the control over the QTM RT interface. See TakeControl.

Response

The command will return a command packet or an error packet.

```
Response: 'Measurement loaded'

Errors:   'Missing file name'
          'Failed to load measurement'
          'Active camera connection exists'
          'Current measurement not saved'
```

```
                'Parse error'
                'You must be master to issue this command'
```

Save

```
Save Filename ['Overwrite']
```

Filename: A string containing the name of the file to save the current measurement to. If the filename doesn't end with ".qtm", it will be added to the end of the filename. The file name can be a relative or absolute path. See below.

Overwrite: If this parameter is present, an existing measurement with the same name will be overwritten. Otherwise a file exists error response will be sent. This parameter is optional.

This command will save the current measurement to file. The name of the file is given in the argument. The file name can be relative or absolute. If the file name is relative, QTM will save the file in the data folder located in the project folder. If the file already exists, it will be overwritten if the Overwrite parameter is present. Otherwise a counter will be added to the end of the file name (_##). If the filename includes spaces, the whole filename should be enclosed by quotation marks.

It is only possible to issue this command if you have the control over the QTM RT interface. See TakeControl.

Response

The command will return a command packet or an error packet.

```
Responses: 'Measurement saved'
           'Measurement saved as' Filename

Errors:    'Failed to save measurement'
           'No write access'
           'Failed to create directory'
           'Bad filename'
           'No measurement to save'
           'Active camera connection exists'
           'Parse error'
           'You must be master to issue this command'
```

LoadProject

```
LoadProject ProjectPath
```

ProjectPath: A string containing the path of the project to load.

This command will load a project, given a project path. If the path doesn't exist, current measurement isn't saved or an active camera connection exists, the project will not load.

It is only possible to issue this command if you have the control over the QTM RT interface. See TakeControl.

Response

The command will return a command packet or an error packet.

```
Response: 'Project loaded'

Errors:   'Missing project name'
          'Failed to load project'
          'Active camera connection exists'
          'Current measurement not saved'
          'Parse error'
          'You must be master to issue this command'
```

GetCaptureC3D

```
GetCaptureC3D
```

This command will download the latest capture as a C3D file. If the command is successful, a `Sending capture` response is sent, followed by a C3D file packet containing current capture.

Response

The command will return a command packet and a c3d packet or an error packet.

```
Response: 'Sending capture'

Errors:   'No capture to get'
          'Error sending C3D file'
          'Parse error'
```

GetCaptureQTM

```
GetCaptureQTM
```

This command will download the latest capture as a QTM file. If the command is successful, a `Sending capture` response is sent, followed by a QTM file packet containing current capture.

Response

The command will return a command packet and a qtm packet or an error packet.

```
Response: 'Sending capture'

Errors:   'No capture to get'
          'Error sending QTM file'
          'Parse error'
```

Trig

> **`Trig`**

This command will trig a measurement, if the camera system is set to start on external trigger. The RT server will send a WaitingForTrigger event when it is waiting for a trigger. See Events. It is only possible to issue this command if you have the control over the QTM RT interface. See TakeControl.

Response

The command will return a command packet or an error packet.

```
Response: 'Trig ok'

Errors:   'QTM not waiting for trig'
          'Parse error'
          'You must be master to issue this command'
```

SetQTMEvent

> **`SetQTMEvent`** `Label`

Label: A string containing the label name of the event. If no name is given, the label will be set to “Manual event”.

This command will set an event in QTM.

Response

The command will return a command packet or an error packet.

```
Response: 'Event set'

Errors:   'Event label too long'
          'QTM is not capturing'
```

```
           'Parse error'
           'You must be master to issue this command'
```

Reprocess

> **`Reprocess`**

This command will reprocess current measurement. It is only possible to issue this command if you have the control over the QTM RT interface. See TakeControl.

Response

The command will return a command packet or an error packet.

```
Response: 'Reprocessing file'

Errors:    'No file open'
           'RT from file running'
           'Parse error'
           'You must be master to issue this command'
```

Calibrate

> **`Calibrate`** `[Refine]`

Refine:

The 'Refine' argument is optional. It tells QTM to perform a calibration refinement.

This command will start calibration in QTM. It is only possible to issue this command if you have the control over the QTM RT interface. See TakeControl. The server will send a command packet with the string `Starting calibration` if calibration was started. The command will not wait for the calibration to finish before responding. Wait for the 'Calibration Stopped' event.

After the "Calibration Stopped" event QTM will send an [XML calibration packet](#Calibration XML parameters) to all connected clients.

Response

The command will return a command packet or an error packet.

```
Response: 'Starting calibration'

Errors:    'Can not start calibration'
```

```
            'Parse error'
            'You must be master to issue this command'
```

Led

> **Led** camera mode color

camera: Number of the Miqus camera to change the LED.

mode: This can be one of `On` , `Off` or `Pulsing` .

color: This can be one of `Green` , `Amber` or `All` .

This command can turn the leds on a Miqus camera on/off. You can specify if the Miqus leds should be on, off or pulsing in all or individual colors (green, amber).

Response

The command can return an error packet.

```
Errors:    'Camera system not running'
           'Parse error'
           'You must be master to issue this command'
```

Quit

> **Quit**

This command ends the current telnet session. The Quit command only works if you have connected to the RT server on the telnet port. Default telnet port is 22221.

Response

The command will return a command packet.

```
Response: 'Bye bye'
```

QTM RT Packets.

Structure

All packets sent to or from the server have the same general layout.

The first part consists of a packet header of 8 bytes:

BYTES	NAME	TYPE	DESCRIPTION
4	Size	32-bit integer	The total size of the QTM RT packet including these four bytes denoting the size.
4	Type	32-bit integer	The type of data in the packet

After the header follows the actual data of the packet:

BYTES	NAME	TYPE	DESCRIPTION
Size - 8	Data	Mixed	Whatever data that the Type field says it is.

Please note: A packet sent to or from a QTM RT server is not a type of TCP data packet. TCP is defined as a data stream. QTM RT server data packets are part of the QTM RT server protocol defined on top of a TCP stream. When a client reads data from the TCP/IP stream, it is usually divided into chunks (each probably being sent in a single TCP/IP packet), but these chunks are not necessarily the same as a QTM RT server protocol packet. To handle TCP/IP reading properly, first read four bytes from the stream to see how big the packet is, then read (Size – 4) bytes from the TCP/IP stream to make sure you have received a whole packet. Then handle the packet according to its Type member.

Packet types

The Type field of a QTM RT server packet header is a number that should be interpreted according to the table below. These are the data types that are defined in the protocol so far. Detailed descriptions of the data packets for each type can be found in the sections following this one.

TYPE NO	NAME	DESCRIPTION
0	Error	The last command generated an error. The error message is included in the packet.
1	Command	A command sent to the server or a response from the server to a command indicating that the command was successful.
2	XML	Data sent by the server in the form of XML, or data sent to the server in the form of XML.
3	Data	One sample of real-time data sent from the server. The contents of the frame may vary depending on the commands/settings sent to the server. The contents may also vary between frames due to different sampling frequencies and buffering properties of different data types.
4	No More Data	This packet type contains no data. It is a marker used to indicate that a measurement has finished or is not yet started.

5	C3D file	Data sent from the server in form of a C3D file.
6	Event	This packet type contains event data from QTM.
7	Discover	Auto discover packet.
8	QTM file	Data sent from the server in form of a QTM file.

Error packet

Error messages from the server are sent in an error packet. Whenever you read a response from the server, it may be an error packet instead of the packet type you expect. Command packet strings sent from the server are always NULL-terminated.

Example of an error packet:

BYTES	NAME	VALUE
4	Size	31 (8 bytes header + 23 bytes data)
4	Type	0
23	Data	"Command not supported."

Command packet

Commands and responses to commands are sent in packets of type 1. Command packets sent from the server always contains NULL-terminated strings. However, NULL-termination is optional for command strings sent from the clients to the server.

Here is an example of a command sent to the server:

BYTES	NAME	VALUE
4	Size	20 (8 bytes header + 12 bytes data)
4	Type	1
12	Data	"Version 1.2"

XML packet

XML is used to exchange settings parameters between the server and the client. XML packets follow the same layout as Command packets and Error packets. The packet header is followed by a `NULL` -terminated ASCII string. All XML data strings sent from the QTM RT server are enclosed by a element named from the version of the protocol used (`QTM_Parameters_Ver_1.22` in this version of the protocol).

When requesting more than one type of parameters at the same time, all of them are placed in the same `QTM_Parameters_Ver_1.22` element. The individual elements may appear in any order inside this element.

BYTES	NAME	VALUE
4	Size	8 bytes header + XML string length
4	Type	2
	Data	XML string data, NULL terminated. The XML data can consist of one or several of following parameters: General, 3D, 6D, GazeVector, EyeTracker, Analog, Force, Image and Skeleton.

Following settings can be changed in QTM by sending an XML packet. General, 6D, Force, Image and Skeleton. The packet must start with an element called `QTM_Settings` .

General XML parameters

In response to the command `GetParameters General` the QTM RT server will reply with an XML data packet, containing an element called General. See below for the format of this element.

Changing parameters

To change the General settings in QTM, send an XML data packet containing a General element.

XML Format

```
<General>
    <Frequency></Frequency>
    <Capture_Time></Capture_Time>
    <Start_On_External_Trigger></Start_On_External_Trigger>
    <Start_On_Trigger_NO></Start_On_Trigger_NO>
    <Start_On_Trigger_NC></Start_On_Trigger_NC>
    <Start_On_Trigger_Software></Start_On_Trigger_Software>
    <External_Time_Base>
        <Enabled></Enabled>
        <Signal_Source></Signal_Source>
        <Signal_Mode></Signal_Mode>
        <Frequency_Multiplier></Frequency_Multiplier>
        <Frequency_Divisor></Frequency_Divisor>
        <Frequency_Tolerance></Frequency_Tolerance>
        <Nominal_Frequency></Nominal_Frequency>
        <Signal_Edge></Signal_Edge>
        <Signal_Shutter_Delay></Signal_Shutter_Delay>
        <Non_Periodic_Timeout></Non_Periodic_Timeout>
    </External_Time_Base>
    <External_Timestamp>
        <Enabled></Enabled>
        <Type></Type>
        <Frequency></Frequency>
    </External_Timestamp>
    <Processing_Actions>
        <PreProcessing2D></PreProcessing2D>
        <Tracking></Tracking>
        <TwinSystemMerge></TwinSystemMerge>
        <SplineFill></SplineFill>
        <AIM></AIM>
        <Track6DOF></Track6DOF>
        <SkeletonSolve></SkeletonSolve>
        <ForceData></ForceData>
        <GazeVector></GazeVector>
        <ExportTSV></ExportTSV>
        <ExportC3D></ExportC3D>
        <ExportMatlabFile></ExportMatlabFile>
        <ExportAviFile></ExportAviFile>
        <ExportFbx></ExportFbx>
        <StartProgram></StartProgram>
    </Processing_Actions>
    <RealTime_Processing_Actions>
        <PreProcessing2D></PreProcessing2D>
        <Tracking></Tracking>
        <AIM></AIM>
        <Track6DOF></Track6DOF>
        <SkeletonSolve></SkeletonSolve>
        <ForceData></ForceData>
        <GazeVector></GazeVector>
    </RealTime_Processing_Actions>
    <Reprocessing_Actions>
        <PreProcessing2D></PreProcessing2D>
        <Tracking></Tracking>
        <TwinSystemMerge></TwinSystemMerge>
        <SplineFill></SplineFill>
        <AIM></AIM>
        <Track6DOF></Track6DOF>
        <SkeletonSolve></SkeletonSolve>
        <ForceData></ForceData>
        <GazeVector></GazeVector>
        <ExportTSV></ExportTSV>
        <ExportC3D></ExportC3D>
        <ExportMatlabFile></ExportMatlabFile>
        <ExportAviFile></ExportAviFile>
        <ExportFbx></ExportFbx>
        <StartProgram></StartProgram>
    </Reprocessing_Actions>
    <EulerAngles First Second Third/>
    <Camera>
        <ID></ID>
        <Model></Model>
        <Underwater></Underwater>
        <Supports_HW_Sync></Supports_HW_Sync>
        <Serial></Serial>
        <Mode></Mode>
        <Video_Frequency></Video_Frequency>
        <Video_Resolution></Video_Resolution>
        <Video_Aspect_Ratio></Video_Aspect_Ratio>
        <Video_Exposure>
            <Current></Current>
            <Min></Min>
            <Max></Max>
        </Video_Exposure>
        <Video_Flash_Time>
            <Current></Current>
            <Min></Min>
            <Max></Max>
        </Video_Flash_Time>
        <Marker_Exposure>
            <Current></Current>
            <Min></Min>
            <Max></Max>
        </Marker_Exposure>
        <Marker_Threshold>
            <Current></Current>
            <Min></Min>
            <Max></Max>
        </Marker_Threshold>
        <Position>
            <X></X>
            <Y></Y>
            <Z></Z>
            <Rot_1_1></Rot_1_1>
            <Rot_2_1></Rot_2_1>
            <Rot_3_1></Rot_3_1>
            <Rot_1_2></Rot_1_2>
            <Rot_2_2></Rot_2_2>
            <Rot_3_2></Rot_3_2>
            <Rot_1_3></Rot_1_3>
            <Rot_2_3></Rot_2_3>
            <Rot_3_3></Rot_3_3>
        </Position>
        <Orientation></Orientation>
        <Marker_Res>
            <Width></Width>
            <Height></Height>
        </Marker_Res>
        <Video_Res>
            <Width></Width>
            <Height></Height>
        </Video_Res>
        <Marker_FOV>
            <Left></Left>
            <Top></Top>
            <Right></Right>
            <Bottom></Bottom>
        </Marker_FOV>
        <Video_FOV>
            <Left></Left>
            <Top></Top>
            <Right></Right>
            <Bottom></Bottom>
        </Video_FOV>
        <Sync_Out>
            <Mode></Mode>
            <Value></Value>
```

```
                <Duty_Cycle></Duty_Cycle>
                <Signal_Polarity></Signal_Polarity>
            </Sync_Out>
            <Sync_Out2>
                <Mode></Mode>
                <Value></Value>
                <Duty_Cycle></Duty_Cycle>
                <Signal_Polarity></Signal_Polarity>
            </Sync_Out2>
            <Sync_Out_MT>
                <Signal_Polarity></Signal_Polarity>
            </Sync_Out_MT>
            <LensControl>
                <Focus Value Min Max/>
                <Aperture Value Min Max/>
            </LensControl>
            <AutoExposure Enabled Compensation/>
            <AutoWhiteBalance></AutoWhiteBalance>
        </Camera>
    </General>
```

Frequency

Element containing the QTM capture frequency. Integer value.

Capture_Time

Element containing the length of the QTM capture, started with the start command. Capture_Time is a float, expressed in seconds.

Start_On_External_Trigger

Element containing true if measurement starts on external trigger, else false.

Start_On_Trigger_NO

Element containing true if measurement start on external trigger signal from a Sync Unit Trig NO port or the Oqus trigger input, else false.

Start_On_Trigger_NC

Element containing true if measurement start on external trigger signal from a Sync Unit Trig NC port, else false.

Start_On_Trigger_Software

Element containing true if measurement starts on software trigger, else false. Software trigger can come from devices and applications like keyboard, RT clients, telnet command etc.

External_Time_Base

Element containing external time base information.

ELEMENT	DESCRIPTION	TYPE
Enabled	Enable/disable external time base.	True or False
Signal_Source	Signal source for external time base.	Control port, IR receiver, SMPTE, IRIG, Video sync
Signal_Mode	Signal mode for external time base.	Periodic or Non-periodic
Frequency_Multiplier	Multiplier that is applied to incoming frequency to get the camera frequency. Can be combined with frequency divisor.	*integer*
Frequency_Divisor	Divisor that is applied to incoming frequency to get the camera frequency. Can be combined with frequency multiplier.	*integer*
Frequency_Tolerance	frequency tolerance in ppm of period time.	*integer*
Nominal_Frequency	Nominal frequency used by QTM. If the value is None, nominal frequency is disabled. Otherwise the value is a float.	None or *float*
Signal_Edge	Control port TTL signal edge.	Negative or Positive
Signal_Shutter_Delay	Delay from signal to shutter opening in micro seconds.	*integer*
Non_Periodic_Timeout	Max number of seconds expected between two frames in non-periodic mode.	*float*

External_Timestamp

The External_Timestamp element contains following elements.

ELEMENT	DESCRIPTION	TYPE
Enabled	Enable/disable external timestamp.	True or False
Type	External timestamp type.	SMPTE, IRIG or CameraTime
Frequency	Frequency used by external timestamp.	*integer*

Processing_Actions

The Processing_Actions element contains following elements.

ELEMENT	DESCRIPTION	TYPE
PreProcessing2D	2D pre-processing.	True or False
Tracking	2D or 3D tracking.	2D, 3D or False
TwinSystemMerge	Twin system merge.	True or False

SplineFill	Spline fill.	True or False
AIM	AIM.	True or False
Track6DOF	6DOF tracking.	True or False
ForceData	Force	True or False
GazeVector	Gaze vector.	True or False
SkeletonSolve	Skeleton solving.	True or False
ExportTSV	Export to TSV file.	True or False
ExportC3D	Export to C3D file.	True or False
ExportMatlabFile	Export to MATLAB file.	True or False
ExportAviFile	Export to AVI file.	True or False
ExportFBX	Export to FBX file.	True or False
StartProgram	Start an external program.	True or False

RealTime_Processing_Actions

The RealTime_Processing_Actions element contains following elements.

ELEMENT	DESCRIPTION	TYPE
PreProcessing2D	2D pre-processing.	True or False
Tracking	2D or 3D tracking.	3D or False
AIM	AIM.	True or False
Track6DOF	6DOF tracking.	True or False
ForceData	Force	True or False
GazeVector	Gaze vector.	True or False
SkeletonSolve	Skeleton solving.	True or False

Reprocessing_Actions

The Reprocessing_Actions element contains following elements.

ELEMENT	DESCRIPTION	TYPE
PreProcessing2D	2D pre-processing.	True or False
Tracking	2D or 3D tracking.	2D, 3D or False
TwinSystemMerge	Twin system merge.	True or False
SplineFill	Spline fill.	True or False

AIM	AIM.	True or False
Track6DOF	6DOF tracking.	True or False
ForceData	Force	True or False
GazeVector	Gaze vector.	True or False
SkeletonSolve	Skeleton solving.	True or False
ExportTSV	Export to TSV file.	True or False
ExportC3D	Export to C3D file.	True or False
ExportMatlabFile	Export to MATLAB file.	True or False
ExportAviFile	Export to AVI file.	True or False
ExportFBX	Export to FBX file.	True or False
StartProgram	Start an external program.	True or False

EulerAngles

EulerAngles element contains three attributes.

ATTRIBUTE	DESCRIPTION
First	Name of first Euler rotation.
Second	Name of second Euler rotation.
Third	Name of third Euler rotation.

Camera

General settings consist of none or several Camera elements.

ELEMENT	DESCRIPTION	TYPE
ID	Identity of the camera to which the settings apply.	*integer*
Model	Camera model. For available camera models see Camera Models.	*string*
Underwater	Camera is an underwater camera.	True or False
Supports_HW_Sync	Camera supports hardware sync.	True or False
Serial	Camera serial number	*string*
Mode	Camera mode.	Marker, Marker intensity, Video

Video_Frequency	Video capture frequency.	*integer*
Video_Resolution	Video resolution for non-marker cameras (Oqus 2c, Miqus Video and Miqus Hybrid).	1080p, 720p, 540p, 480p
Video_Aspect_Ratio	Aspect ratio for non-marker cameras (Oqus 2c, Miqus Video and Miqus Hybrid).	16x9, 4x3, 1x1
Video_Exposure	Contains elements: Current, Min, Max. Values are in micro seconds.	*integer*
Video_Flash_Time	Contains elements: Current, Min, Max. Values are in micro seconds.	*integer*
Marker_Exposure	Contains elements: Current, Min, Max. Values are in micro seconds.	*integer*
Marker_Threshold	Contains elements: Current, Min, Max. Values are in micro seconds.	*integer*
Position	Camera position and rotation. Position elements: X, Y, Z. Rotation matrix elements: Rot_1_1, Rot_2_1, Rot_3_1, Rot_1_2, Rot_2_2, Rot_3_2, Rot_1_3, Rot_2_3, Rot_3_3.	*float*
Orientation	QTM 2D camera view orientation. Possible values are 0, 90, 180, 270 degrees.	*integer*
Marker_Res	Camera marker resolution. Contains elements Width and Height. Values are in sub pixels.	*integer*
Video_Res	Camera video resolution. Contains elements Width and Height. Values are in pixels.	*integer*
Marker_FOV	Camera marker field of view. Contains elements Left, Top, Right and Bottom. Values are in pixels.	*integer*
Video_FOV	Camera video field of view. Contains elements Left, Top, Right and Bottom. Values are in pixels.	*integer*
Sync_Out	Contains elements: Mode, Value, Duty_Cycle, Signal_Polarity. See Sync Out Element.	
Sync_Out2	Contains elements: Mode, Value, Duty_Cycle, Signal_Polarity. See Sync Out Element.	
Sync_Out_MT	Contains element Signal_Polarity.	Negative or Positive
LensControl	Contains elements Focus and Aperture with attributes: Value, Min, Max.	*float*
AutoExposure	Contains attributes: Enabled (bool) and Compensation (float).	True or False, *float*
AutoWhiteBalance	Enable auto white balance.	True or

		False

Camera Models

- MacReflex
- ProReflex 120
- ProReflex 240
- ProReflex 500
- ProReflex 1000
- Oqus 100
- Oqus 200 C
- Oqus 300
- Oqus 300 Plus
- Oqus 400
- Oqus 500
- Oqus 500 Plus
- Oqus 600 Plus
- Oqus 700
- Miqus M1
- Miqus M3
- Miqus M5
- Miqus Hybrid
- Miqus Sync Unit
- Miqus Video
- Miqus Video Color
- Arqus A5
- Arqus A9
- Arqus A12
- Arqus A26

Sync Out Element

This element is part of Sync_Out and Sync_Out2 elements.

ELEMENT	DESCRIPTION	TYPE
Mode	Sync out mode.	Shutter out, Multiplier, Divisor, Camera independent, Continuous 100Hz
Value	value only used for sync out modes: Multiplier, divisor or camera independent value.	*integer*
Duty_Cycle	Duty cycle only used for sync out modes: Multiplier, divisor or camera independent value.	*float*
Signal_Polarity	Not used for continuous 100Hz.	Negative or Positive

3D XML parameters

In response to the command `GetParameters 3D` the QTM RT server will reply with an XML data packet, containing an element called The_3D. See below for the format of this element.

XML Format

```
<The_3D>
    <AxisUpwards></AxisUpwards>
    <CalibrationTime></CalibrationTime>
    <Labels></Labels>
    <Label>
        <Name></Name>
        <RGBColor></RGBColor>
        <Trajectory_Type></Trajectory_Type>
    </Label>
    <Bones>
        <Bone From To Color/>
    </Bones>
</The_3D>
```

AxisUpwards

This element tells which axis that is pointing upwards in QTM. The value can be one of following: +X, +Y, +Z, -X, -Y and -Z.

CalibrationTime

This element tells the date and time of when the system was last calibrated. If the system has no valid calibration the value is empty. The calibration date and time is formatted like this: `yyyy.mm.dd hh:mm:ss` . Example, "2011.09.23 11:23:11"

Labels

Number of Label elements.

Label

Element containing elements Name and RGBColor. There is one Label element for each identified trajectory. The order of the trajectories is the same as in the 3d data packets.

ELEMENT	DESCRIPTION	TYPE
Name	Trajectory name.	*string*
RGBColor	Trajectory color. Represented by a three byte integer value. Bit 0-7 represents red, bit 8-15 represents green and bit 16-23 represents blue.	*integer*
Trajectory_Type	Type of the trajectory. Possible types are: Mixed, Measured, Gap-filled, Virtual, Edited, Measured Slave, Gap-filled Slave, Virtual Slave and Edited Slave.	

Bones

Element containing one or more bone elements.

Bone

Element containing attributes: fromName, toName and Color.

ATTRIBUTE	DESCRIPTION	TYPE
fromName	Label name of the trajectory where the bone starts.	*string*
toName	Label name of the trajectory where the bone ends.	*string*
Color	Bone color. Represented by a three byte integer value. Bit 0-7 represents red, bit 8-15 represents green and bit 16-23 represents blue.	

6D XML parameters

In response to the command `GetParameters 6D` the QTM RT server will reply with an XML data packet, containing the element The_6D.

Changing parameters

To change the current 6d project settings in QTM, send an XML data packet containing a The_6D element. When sending 6d settings to the server, all current rigid bodies will be deleted from the project.

XML Format

```
<The_6D>
    <Body>
        <Name></Name>
        <Color R G B/>
        <MaximumResidual></MaximumResidual>
        <MinimumMarkersInBody></MinimumMarkersInBody>
        <BoneLengthTolerance></BoneLengthTolerance>
        <Filter Preset/>
        <Mesh>
            <Name></Name>
            <Position X Y Z/>
            <Rotation X Y Z/>
            <Scale></Scale>
            <Opacity></Opacity>
        </Mesh>
        <Points>
            <Point X Y Z Virtual PhysicalId Name/>
            <Point X Y Z Virtual PhysicalId Name/>
            <Point X Y Z Virtual PhysicalId Name/>
        </Points>
        <Data_origin X Y Z Relative_body></Data_origin>
        <Data_orientation R11 R12 R13 R21 R22 R23 R31 R32 R33 Relative_body>
        </Data_orientation>
    </Body>
</The_6D>
```

Body

Element containing 6DOF body information.

Name

Element containing the name of the 6DOF body. Name must always be present.

Color

Element containing the color of the 6DOF body, represented by three attributes R, G and B. Each attribute is an integer 0 - 255.

MaximumResidual

Element containing maximal residual of the rigid body. Value is a float.

MinimumMarkersInBody

Element containing minimal number of markers needed to detect the rigid body. Value is an integer.

BoneLengthTolerance

Element containing bone length tolerance for the rigid body. Value is a float.

Filter

Element containing an attribute, Preset. Preset is the name of the filter preset used by the body. See table below for available pre-sets.

FILTER PRE-SET	DESCRIPTION
No filter	Filter disabled.
Multi-purpose	Light smoothing and jitter reduction for multi-purpose use.
High stability	Considerable smoothing and jitter reduction for stabilizing noisy data in large capture volumes. May introduce some lag and overshoot.
Static pose	Effective jitter reduction for rigid bodies in static positions. May introduce some lag for moving bodies.

Mesh

Element contains following elements:

- Name - Name of the obj file defining the mesh.
- Position - Contains attributes X, Y, and Z with coordinate values as floats.
- Rotation - Contains attributes X, Y, and Z with rotation values as floats.
- Scale - Scale of mesh object. Value is a float.
- Opacity - Opacity of mesh object. Value is a float.

Points

Element contains 3 or more Point elements.

Point

Element contains attributes: X, Y, Z, Virtual, PhysicalId and Name.

- X - Coordinate of point in body. Value is a float.
- Y - Coordinate of point in body. Value is a float.
- Z - Coordinate of point in body. Value is a float.
- Virtual - Point is virtual if value is 1. Value is 0 for non virtual point.
- PhysicalId - Physical id of point. Value is an integer.
- Name - Name of point.

Data_origin

Element contains origin type. 0 = Global, 1 = Relative and 2 = Fixed. It can also contain attributes, depending on the origin type.

Relative origin

Data_origin contains one attribute, Relative_body. Relative body is the body index of the related body. Index starting on 1.

Fixed Origin

Data_Origin contains attributes: X, Y and Z. The coordinates defines the translation relative the global coordinate system.

Data_orientation

Element contains origin type. 0 = Global, 1 = Relative and 2 = Fixed. It can also contain attributes, depending on the origin type.

Relative origin

Data_orientation contains one attribute, Relative_body. Relative body is the body index of the related body. Index starting on 1.

Fixed Origin

Data_orientation contains attributes: R11, R12, R13, R21, R22, R23, R31, R32 and R33. The attributes defines a 3x3 rotation matrix with the orientation relative the global coordinate system.

Gaze vector XML parameters

In response to the command `GetParameters GazeVector` the QTM RT server will reply with an XML data packet, containing an element called Gaze_Vector. Gaze_Vector can contain several Vectors.

XML Format

```
<Gaze_Vector>
    <Vector>
        <Name></Name>
        <Frequency></Frequency>
        <Hardware_Sync></Hardware_Sync>
        <Filter></Filter>
    </Vector>
</Gaze_Vector>
```

Vector

Element containing gaze vector information.

Name

The name of the gaze vector body.

Frequency

The gaze vector update frequency.

Hardware_Sync

True if gaze vector data is hardware synchronized, else False.

Filter

True if gaze vector data is filtered, else False.

Eye tracker XML parameters

In response to the command `GetParameters EyeTracker` the QTM RT server will reply with an XML data packet, containing an element called Eye_Tracker. Eye_Tracker can contain several Devices.

XML Format

```
<Eye_Tracker>
    <Device>
        <Name></Name>
        <Frequency></Frequency>
        <Hardware_Sync></Hardware_Sync>
    </Device>
</Eye_Tracker>
```

Device

Element containing eye tracker device information.

Name

The name of the eye tracker body.

Frequency

The eye tracker update frequency.

Hardware_Sync

True if eye tracker data is hardware synchronized, else False.

Analog XML parameters

In response to the command `GetParameters Analog` the QTM RT server will reply with XML data packet, containing an element called Analog. Analog can contain several devices.

XML Format

```
<Analog>
    <Device>
        <Device_ID></Device_ID>
        <Device_Name></Device_Name>
        <Channels></Channels>
        <Frequency></Frequency>
        <Range>
            <Min></Min>
            <Max></Max>
        </Range>
        <Channel>
            <Label></Label>
            <Unit></Unit>
        </Channel>
    </Device>
</Analog>
```

Device

Element containing analog device information.

Device_ID

Unique ID of the analog device. An integer value starting with 1.

Device_Name

Analog device name.

Channels

Number of analog channels.

Frequency

Analog measurement frequency.

Range

Min and max analog measurement values.

Channel

Device contains one Channel element per analog channel, containing Label and Unit.

Force XML parameters

In response to the command `GetParameters Force` the QTM RT server will reply with XML data packet, containing an element called Force. Force can contain several Plates.

Changing parameters

To change the Force settings in QTM, send an XML data packet containing a Force element. The only setting that is possible to change is force plate location.

XML Format

```
<Force>
    <Unit_Length></Unit_Length>
    <Unit_Force></Unit_Force>
    <Plate>
        <Force_Plate_Index></Force_Plate_Index>
        <Analog_Device_ID></Analog_Device_ID>
        <Frequency></Frequency>
        <Type></Type>
        <Name></Name>
        <Length></Length>
        <Width></Width>
        <Location>
            <Corner1>
                <X></X>
                <Y></Y>
                <Z></Z>
            <Corner1>
            <Corner2>
                <X></X>
                <Y></Y>
                <Z></Z>
            <Corner2>
            <Corner3>
                <X></X>
                <Y></Y>
                <Z></Z>
            <Corner3>
            <Corner4>
                <X></X>
                <Y></Y>
                <Z></Z>
            <Corner4>
        </Location>
        <Origin>
            <X></X>
            <Y></Y>
            <Z></Z>
        </Origin>
        <Channels>
            <Channel>
                <Channel_No></Channel_No>
```

```
                    <ConversionFactor></ConversionFactor>
                </Channel>
            </Channels>
            <Calibration_Matrix>
                <Rows>
                    <Row>
                        <Columns>
                            <Column></Column>
                        </Columns>
                    </Row>
                </Rows>
            </Calibration_Matrix>
        </Plate>
    </Force>
```

Unit_Length

Length unit used in the Force XML element.

Unit_Force

Force unit used in the Force XML element.

Plate

Element containing force plate information.

Force_Plate_Index

Unique ID of the force plate. An integer value starting at 1. The index is used to identify which plate to change when changing Force settings.

Analog_Device_ID

ID of the analog device connected to the force plate. If the ID is 0, there is no analog device associated with this force plate.

Frequency

Measurement frequency of the analog device connected to the force plate.

Type

Force plate type. Supported force plates: AMTI, AMTI 8 Channels, Bertec, Kistler and QMH.

Name

Force plate name.

Length

Force plate length.

Width

Force plate width.

Location

Element containing four elements with the corners of the force plate. Corner1, Corner2, Corner3 and Corner4. Each corner has an X, Y and Z coordinate value. This setting is possible to change.

Origin

Element containing X, Y and Z coordinates for the force plate origin.

Channels

Element containing elements called Channel. One for each analog channel connected to the force plate. Each Channel contains Channel_No and ConversionFactor.

Calibration_Matrix

Element containing a 6x6, 6x8 or 12x12 calibration matrix for the force plate.

Image XML parameters

In response to the command `GetParameters Image` the QTM RT server will reply with XML data packet, containing an element called Image. Image can contain several cameras.

Changing parameters

To change the Image settings in QTM, send an XML data packet containing an Image element.

XML Format

```
<Image>
    <Camera>
        <ID></ID>
        <Enabled></Enabled>
        <Format></Format>
        <Width></Width>
        <Height></Height>
        <Left_Crop></Left_Crop >
        <Top_Crop></Top_Crop>
        <Right_Crop></Right_Crop>
        <Bottom_Crop></Bottom_Crop>
    </Camera>
</Image>
```

ID

Camera ID for the camera to which the settings apply.

Enabled

Turn on or of Image streaming from the selected camera.

Format

Picture format of the image stream. Available formats are: `RAWGrayscale` , `RAWBGR` , `JPG` and `PNG` .

Width

Width of the streaming image. This does not take into account the cropping. The width is the dimensions had the image not been cropped at all. Note that this does not have to be the same as the requested width, due to scaling in QTM. 32-bit integer.

Height

Height of the streaming image. This does not take into account the cropping. The height is the dimensions had the image not been cropped at all. Note that this does not have to be the same as the requested width, due to scaling in QTM.

Left_Crop

Position of the requested image left edge relative the original image. 32-bit float.

0.0 = Original image left edge **(Default)**. 1.0 = Original image right edge.

Top_Crop

Position of the requested image top edge relative the original image. 32-bit float.

0.0 = Original image top edge **(Default)**. 1.0 = Original image bottom edge.

Right_Crop

Position of the requested image right edge relative the original image. 32-bit float.

0.0 = Original image left edge.
1.0 = Original image right edge **(Default)**.

Bottom_Crop

Position of the requested image bottom edge relative the original image. 32-bit float.

0.0 = Original image top edge.
1.0 = Original image bottom edge **(Default)**.

Skeleton XML parameters

In response to the command `GetParameters Skeleton` the QTM RT server will reply with an XML data packet, containing an element called Skeletons. Skeletons can contain several skeletons.

Changing parameters

To change the current Skeleton project settings in QTM, send an XML data packet containing an Skeletons element. When sending skeleton settings to the server, all current skeleton definitions will be deleted from the project.

XML Format

```
<Skeletons>
    <Skeleton Name>
        <Scale></Scale>
        <Segments>
            <Segment Name ID>
                <Solver></Solver>
                <Transform>
                    <Position X Y Z/>
                    <Rotation X Y Z W/>
                </Transform>
                <DefaultTransform>
                    <Position X Y Z/>
                    <Rotation X Y Z W/>
                </DefaultTransform>
                <DegreesOfFreedom>
                    <RotationX>
                        <Constraint LowerBound UpperBound/>
                        <Couplings>
                            <Coupling Segment DegreeOfFreed
om Coefficient/>
                        </Couplings>
                        <Goal Value Weight/>
                    </RotationX>
                    <RotationY>
                        <Constraint LowerBound UpperBound/>
                        <Couplings>
                            <Coupling Segment DegreeOfFreed
om Coefficient/>
                        </Couplings>
                        <Goal Value Weight/>
                    </RotationY>
                    <RotationZ>
                        <Constraint LowerBound UpperBound/>
                        <Couplings>
                            <Coupling Segment DegreeOfFreed
om Coefficient/>
                        </Couplings>
                        <Goal Value Weight/>
                    </RotationZ>
                    <TranslationX>
                        <Constraint LowerBound UpperBound/>
                        <Couplings>
                            <Coupling Segment DegreeOfFreed
om Coefficient/>
                        </Couplings>
                        <Goal Value Weight/>
                    </TranslationX>
                    <TranslationY>
                        <Constraint LowerBound UpperBound/>
                        <Couplings>
                            <Coupling Segment DegreeOfFreed
```

```
om Coefficient/>
                        </Couplings>
                        <Goal Value Weight/>
                    </TranslationY>
                    <TranslationZ>
                        <Constraint LowerBound UpperBound/>
                        <Couplings>
                            <Coupling Segment DegreeOfFreed
om Coefficient/>
                        </Couplings>
                        <Goal Value Weight/>
                    </TranslationZ>
                </DegreesOfFreedom>
                <Endpoint X Y Z/>
                <Markers>
                    <Marker Name>
                        <Position X Y Z/>
                        <Weight></Weight>
                    </Marker>
                </Markers>
                <RigidBodies>
                    <RigidBody Name>
                        <Transform>
                            <Position X Y Z/>
                            <Rotation X Y Z W/>
                        </Transform>
                        <Weight></Weight>
                    </RigidBody>
                </RigidBodies>
                <Segment></Segment> <!-- Child segments. Ca
n be empty. -->
            </Segment>
        </Segments>
    </Skeleton>
</Skeletons>
```

Skeleton

The Skeleton element can contain several Skeleton elements. The Skeleton element contains an attribute called Name, with the name of the skeleton. The skeleton element contains following elements: Scale and Segments.

ATTRIBUTE	DESCRIPTION
Name	Skeleton name.

Scale

The Scale element contains the skeleton scale factor.

Segments

The Segments element contains one Segment element, the root segment.

Segment

The Segment element contains two attributes: Name and ID. The id is used to identify the skeleton segments streamed by the QTM real-time server. The segment element contains following elements: Solver, Transform, DefaultTransform, DegreesOfFreedom, Endpoint, Markers and Segment. There can be more than one Segment element, or none.

ATTRIBUTE	DESCRIPTION
Name	Segment name.
ID	Segment id, used to identify the skeleton segments streamed by the QTM real-time server. To get valid segment IDs, you have to be connected to the camera system or running rt from file when reading skeleton settings. The ID is not used when sending skeleton settings to QTM.

Solver

The Solver element contains the name of the solver used. Available solvers: Global Optimization.

Transform

The Transform element contains two elements: Position and Rotation.

Position

Position element contains following attributes.

ATTRIBUTE	DESCRIPTION
X	Segment X position.
Y	Segment Y position.
Z	Segment Z position.

Rotation

Rotation element contains following attributes.

ATTRIBUTE	DESCRIPTION
X	Segment X rotation.
Y	Segment Y rotation.
Z	Segment Z rotation.
W	Segment W rotation.

DefaultTransform

The DefaultTransform element contains two elements: Position and Rotation. The elements are the same as format as the elements in Transform.

DegreesOfFreedom

DegreesOfFreedom can contain following elements: RotationX, RotationY, RotationZ, TranslationX, TranslationY, TranslationZ.

RotationX, RotationY, RotationZ, TranslationX, TranslationY, TranslationZ

All rotation and translation elements contain following elements: Constraint, Couplings and Goal.

Constraint

Constraint contains following attributes: LowerBound and UpperBound.

ATTRIBUTE	DESCRIPTION
LowerBound	Segment degrees of freedom lower bound.
UpperBound	Segment degrees of freedom upper bound.

Couplings

Couplings can contain several Coupling elements.

Coupling

Coupling contains following attributes: Segment, DegreeOfFreedom and Factor.

ATTRIBUTE	DESCRIPTION
Segment	Segment name.
DegreeOfFreedom	"RotationX", "RotationY", "RotationZ", "TranslationX", "TranslationY" or "TranslationZ"
Factor	Coupling factor.

Goal

Goal contains following attributes: Value and Weight.

ATTRIBUTE	DESCRIPTION
Value	Goal value.
Weight	Goal weight.

Endpoint

Endpoint element contains attributes below.

ATTRIBUTE	DESCRIPTION
X	Segment endpoint X position.
Y	Segment endpoint Y position.
Z	Segment endpoint Z position.

Markers

Markers element contains Marker elements.

Marker

Marker element contains attribute Name, with the name of the marker. It also contains two elements: Position and Weight.

Position

The Position element contains attributes with position of the marker. The attributes are the same as Position, used in Transform element.

Weight

Weight element contains the marker weight used by the solver.

RigidBodies

RigidBodies element contains RigidBody elements.

RigidBody

RigidBody element contains attribute Name, with the name of the rigid body. It also contains two elements: Transform and Weight. Transform element has the same format as in the beginning of the Segment element.

Transform

The Transform element has the same format as Transform in the beginning of the Segment element. It contains position and rotation of the rigid body.

Weight

Weight element contains the rigid body weight used by the solver.

Calibration XML parameters

In response to the command `GetParameters Calibration` the QTM RT server will reply with XML data packet, containing an element called Calibration. See below for the format of this element.

XML Format

```
<calibration calibrated source created qtm-version type refit-residual wandLength maximumFrames shortArmEnd longArmEnd longArmMiddle>
    <results std-dev min-max-diff refit-residual consecutive/>
    <cameras>
        <camera active calibrated message point-count avg-residual serial model viewrotation>
            <fov_marker left top right bottom/>
            <fov_marker_max left top right bottom/>
            <fov_videor left top right bottom/>
            <fov_video_max left top right bottom/>
            <transform x y x r11 r12 r13 r21 r22 r23 r31 r32 r33/>
            <intrinsic focalLength sensorMinU sensorMaxU sensorMinV sensorMaxV focalLengthU focalLengthV centerPointU centerPointV skew radialDistortion1 radialDistortion2 radia
```

```
lDistortion3 tangentalDistortion1 tangentalDistortion2/>
        </camera>
    </cameras>
</calibration>
```

calibration

The calibration element contains following attributes.

ATTRIBUTE	DESCRIPTION
calibrated	"true" or "false"
source	Calibration file name.
created	Time and date of calibration.
qtm-version	QTM version used in calibration.
type	Calibration type. "regular","fixed" or "refine".
refit-residual	Wand refit residual. **Only for refine calibration.**
wandLength	Calibration wand length in mm. **Only for regular and refine calibration.**
maximumFrames	Maximum number of frames used for calibration. **Only for regular and refine calibration.**
shortArmEnd	Distance between origin marker and short arm end marker in mm. **Only for regular and refine calibration.**
longarm End	Distance between origin marker and long arm end marker in mm. **Only for regular and refine calibration.**
longarm Middle	Distance between origin marker and long arm middle marker in mm. **Only for regular and refine calibration.**

results

The results element contains following attributes.

ATTRIBUTE	DESCRIPTION
std-dev	Standard deviation of wand length. **Only for regular and refine calibration.**
min-max-diff	Min max diff of wand length. **Only for regular and refine calibration.**
refit-residual	Wand refit residual. **Only for refine calibration.**
consecutive	Consecutive calibrations without reference. **Only for refine calibration.**

cameras

The cameras element contains camera elements.

camera

ATTRIBUTE	DESCRIPTION
active	1 if camera is used in calibration, else 0.
calibrated	"true" or "false"
message	Contains information on failed calibration.
point-count	3d points used in calibration.
avg-residual	Average camera residual in mm.
serial	Serial number of camera.
model	Camera model.
viewrotation	Camera rotation in degrees.

fov-marker, fov-marker-max, fov-video, fov-video-max

The fov elements contain following attributes.

ATTRIBUTE	DESCRIPTION
left	Left most sensor coordinate in pixels.
top	Top most sensor coordinate in pixels.
right	Right most sensor coordinate in pixels.
bottom	Bottom most sensor coordinate in pixels.

transform

The transform element contains following attributes.

ATTRIBUTE	DESCRIPTION
x	Camera x position in mm.
y	Camera y position in mm.
z	Camera z position in mm.
r11	Rotation matrix row 1, col 1.
r12	Rotation matrix row 1, col 2.
r13	Rotation matrix row 1, col 3.
r21	Rotation matrix row 2, col 1.

r22	Rotation matrix row 2, col 2.
r23	Rotation matrix row 2, col 3.
r31	Rotation matrix row 3, col 1.
r32	Rotation matrix row 3, col 2.
r33	Rotation matrix row 3, col 3.

intrinsic

The intrinsic element contains following Attributes.

ATTRIBUTE	DESCRIPTION
focalLength	Focal length in mm
sensorMinU	Sensor width min in sub pixels.
sensorMaxU	Sensor width max in sub pixels.
sensorMinV	Sensor height min in sub pixels.
sensorMaxV	Sensor height max in sub pixels.
focalLengthU	Horizontal focal length in sub pixels.
focalLengthV	Vertical focal length in sub pixels.
centerPointU	Center point in sub pixels.
centerPointV	Center point in sub pixels.
skew	Skew
radialDistortion1	Radial distortion 1
radialDistortion2	Radial distortion 2
radialDistortion3	Radial distortion 3
tangentalDistortion1	Tangential distortion 1
tangentalDistortion2	Tangential distortion 2

Data packet

Each data frame is made up of one or more components, as specified in the commands GetCurrentFrame or StreamFrames. The data frame contains a Count field that specifies the number of components in the frame. Every component starts with a component header – identical (in layout) to the packet header.

Data packet header

BYTES	NAME	TYPE	VALUE/DESCRIPTION

4	Size	32-bit integer	8 bytes packet header + 12 bytes data frame header + the size of all the components and their headers.
4	Type	32-bit integer	Value = 3.
8	Marker Timestamp	64-bit integer	Number of microseconds from start. The timestamp is only valid if at least one camera is in marker mode. The timestamp value is not valid for the Analog, Force and Gaze Vector data frame components, they have their own timestamps in their component data.
4	Marker Frame Number	32-bit integer	The number of this frame. The frame number is only valid if at least one camera is in marker mode. The frame number is not valid for the Analog, Force and Gaze Vector data frame components. They have their own frame numbers within the component.
4	Component Count	32-bit integer	The number of data components in the data packet.

Component data (Repeated *Component Count* times)

BYTES	NAME	TYPE	VALUE/DESCRIPTION
4	Component Size	32-bit integer	Size of Component Data + 8 bytes component header.
4	Component Type	32-bit integer	The type of the component. Defined in the following section.
Size - 8	Component Data	Mixed	Component-specific data. Defined in Data component types and 2D and 2D linearized component sections.

Data component types

The `Component Type` field of the data component header is a number that should be interpreted according to the table below. These are the data frame component types that are defined in the protocol so far.

NAME	TYPE	DESCRIPTION
2D	7	2D marker data
2D_Linearized	8	Linearized 2D marker data
3D	1	3D marker data
3D_Residuals	9	3D marker data with residuals
3D_No_Labels	2	Unidentified 3D marker data
3D_No_Labels_Residuals	10	Unidentified 3D marker data with residuals
6D	5	6D data - position and rotation matrix
6D_Residuals	11	6D data - position and rotation matrix with residuals

6D_Euler	6	6D data - position and Euler angles
6D_Euler_Residuals	12	6D data - position and Euler angles with residuals
Analog	3	Analog data from available analog devices
Analog_Single	13	Analog data from available analog devices. Only one sample per channel and camera frame. The latest sample is used if more than one sample is available.
Force	4	Force data from available force plates.
Force_Single	15	Force data from available force plates. Only one sample per plate and camera frame. The latest sample is used if more than one sample is available.
Image	14	Image frame from a specific camera. Image size and format is set with the XML settings, see Image settings.
Gaze_Vector	16	Gaze vector data defined by a unit vector and position.
Timecode	17	IRIG or SMPTE timecode
Skeleton	18	Skeleton segment information
EyeTracker	19	Eye tracker data with pupil size.

2D components

There are two different 2D components that shares the same marker header.

- 2D
- 2D lineraized

The 2D and 2D linearized data frame format are the same. The only difference is that the coordinates are linearized in 2D linearized.

BYTES	NAME	TYPE	DESCRIPTION
4	Component Size	32-bit integer	The size of the component including the header (Component Size, Component Type and Camera Count).
4	Component Type	32-bit integer	Value 7 or 8. See Data component types.
4	Camera Count	32-bit integer	Number of cameras.
2	2D Drop Rate	16-bit integer	Number of individual 2D frames that have been lost in the communication between QTM and the cameras. The value is in frames per thousand, over the last 0.5 to 1.0 seconds. Range 0-1000. A high value is a sign that the cameras are set at a frequency that is too high for the current network topology to transmit reliably.

2	2D Out Of Sync Rate	16-bit integer	Number of individual 2D frames in the communication between QTM and the cameras, which have not had the same frame number as the other frames. The value is in frames per thousand over the last 0.5 to 1.0 seconds, Range 0-1000. A high value is a sign that the cameras are set at a frequency that is too high for the cameras to process in real time.

Repeated *Camera Count* times:

BYTES	NAME	TYPE	DESCRIPTION
4	Marker Count	32-bit integer	The number of markers for this camera in this frame.
1	Status Flags	8-bit integer	Bit 1: Too much light enters the camera. Please increase the threshold level or lower the exposure time. If measuring at high frequencies, try to reduce the image size. Bit 2-8: Not used.
12 * Marker Count	2D data	Mixed	2D marker data from the camera, described below:

2D Data, repeated *Marker Count* times:

BYTES	NAME	TYPE	DESCRIPTION
4	X	32-bit integer	X coordinate of the marker.
4	Y	32-bit integer	Y coordinate of the marker.
2	Diameter X	16-bit integer	Marker X size.
2	Diameter Y	16-bit integer	Marker Y size.

3D components

There are four different 3D components that shares the same marker header.

- 3D
- 3D with residuals
- 3D no labels
- 3D no labels with residuals

BYTES	NAME	TYPE	DESCRIPTION
4	Component Size	32-bit integer	The size of the component including the header (Component Size, Component Type and Marker Count).
4	Component Type	32-bit integer	Value = 1, 2, 9 or 10. See Data component types.
4	Marker	32-bit	The number of markers in this frame.

	Count	integer	
2	2D Drop Rate	16-bit integer	Number of individual 2D frames that have been lost in the communication between QTM and the cameras. The value is in frames per thousand, over the last 0.5 to 1.0 seconds. Range 0-1000. A high value is a sign that the cameras are set at a frequency that is too high for the current network topology to transmit reliably.
2	2D Out Of Sync Rate	16-bit integer	Number of individual 2D frames in the communication between QTM and the cameras, which have not had the same frame number as the other frames. The value is in frames per thousand over the last 0.5 to 1.0 seconds, Range 0-1000. A high value is a sign that the cameras are set at a frequency that is too high for the cameras to process in real time.

For the *3D* and the *3D with residuals* components, The markers of the 3D data always follow the labels of the 3D parameters. The same number of markers are sent each frame, and in the same order as the labels of the 3D parameters. If a marker is missing from the frame, its X, Y and Z coordinates will have all their 32 bits set - this signifies a negative quiet Not-A-Number according to the IEEE 754 floating point standard.

Repeated *Marker Count* times:

BYTES	NAME	TYPE	DESCRIPTION
4	X	32-bit float	X coordinate of the marker.
4	Y	32-bit float	Y coordinate of the marker.
4	Z	32-bit float	Z coordinate of the marker.
4	ID	32-bit integer	Id that identifies markers between frames. **Only present for 3D no labels and 3D no labels with residuals**.
4	Residual	32-bit float	Residual for the 3D point. **Only present for 3D with residual and 3D no labels with residuals**.

6DOF components

There are four different 6DOF components that shares the same 6dof header.

- 6DOF
- 6DOF with residuals
- 6DOF Euler
- 6DOF Euler with residuals

BYTES	NAME	TYPE	DESCRIPTION

4	Component Size	32-bit integer	The size of the component including the header (Component Size, Component Type and Body Count).
4	Component Type	32-bit integer	Value = 5, 6, 11 or 12. See Data component types.
4	Body Count	32-bit integer	The number of 6DOF bodies in this frame.
2	2D Drop Rate	16-bit integer	Number of individual 2D frames that have been lost in the communication between QTM and the cameras. The value is in frames per thousand, over the last 0.5 to 1.0 seconds. Range 0-1000. A high value is a sign that the cameras are set at a frequency that is too high for the current network topology to transmit reliably.
2	2D Out Of Sync Rate	16-bit integer	Number of individual 2D frames in the communication between QTM and the cameras, which have not had the same frame number as the other frames. The value is in frames per thousand over the last 0.5 to 1.0 seconds, Range 0-1000. A high value is a sign that the cameras are set at a frequency that is too high for the cameras to process in real time.

Repeated *Body Count* times:

BYTES	NAME	TYPE	DESCRIPTION
4	X	32-bit float	X coordinate of the body.
4	Y	32-bit float	Y coordinate of the body.
4	Z	32-bit float	Z coordinate of the body.
9 * 4	Rotation	32-bit float	3x3 rotation matrix of the body. **Only present for 6DOF and 6DOF with residuals**
4	Angle 1	32-bit float	First Euler angle, in degrees, as defined on the Euler tab in QTM's workspace options. **Only present for 6DOF Euler and 6DOF Euler with residuals**
4	Angle 2	32-bit float	Second Euler angle, in degrees, as defined on the Euler tab in QTM's workspace options. **Only present for 6DOF Euler and 6DOF Euler with residuals**

4	Angle 3	32-bit float	Third Euler angle, in degrees, as defined on the Euler tab in QTM's workspace options. **Only present for 6DOF Euler and 6DOF Euler with residuals**
4	Residual	32-bit float	Residual for the 6D body. **Only present for 6DOF with residuals and 6DOF Euler with residuals**

Analog components

There are two different analog components that shares the same analog header.

- Analog
- Analog Single

BYTES	NAME	TYPE	DESCRIPTION
4	Component Size	32-bit integer	The size of the component including the header (Component Size, Component Type and Analog Device Count).
4	Component Type	32-bit integer	Value = 3 or 13. See Data component types.
4	Analog Device Count	32-bit integer	Number of analog devices in this component.

If only streaming a selection of the analog channels, see GetCurrentFrame and StreamFrames, the order of the channels will be the same as in Analog XML parameters.

> The update frequency of the analog data is dependent on the analog data source and its drivers. The QTM real-time server server can only deliver the data at the rate the data source is updated in QTM. Due to this, the analog single data can sometimes return a NaN. This indicates that the server has no updated analog value to transmit. Lower camera frequencies will make it less likely to miss any data.

Analog data

The Analog component sends a packet containing all analog samples that the server has buffered since the last analog frame. It contains it's own sample numbers (one per device), since the analog often runs at different frequency than the camera system.

Repeated *Analog Device Count* times:

BYTES	NAME	TYPE	DESCRIPTION
4	Analog Device	32-bit integer	Id of this analog device.

	ID		
4	Channel Count	32-bit integer	The number of channels of this analog device in this frame.
4	Sample Count	32-bit integer	The number of analog samples per channel in this frame.
4	Sample Number	32-bit integer	Order number of first sample in this frame. Sample Number is increased with the analog frequency. There are Channel Count values per sample number.
4 * Channel Count * Sample Count	Analog Data	32-bit float	All available voltage samples per channel. Example: Channel 1, Sample *Sample Number* Channel 1, Sample *Sample Number* + 1 Channel 1, Sample *Sample Number* + 2 ... Channel 1, Sample *Sample Number* + Sample Count - 1 Channel 2, Sample *Sample Number* Channel 2, Sample *Sample Number + 1* ... Analog Data is omitted if Sample Count is 0.

Analog single data

The Analog single component sends a packet containing only one sample (the latest) per analog channel.

Repeated *Analog Device Count* times:

BYTES	NAME	TYPE	DESCRIPTION
4	Analog Device ID	32-bit integer	Id of this analog device.
4	Channel Count	32-bit integer	The number of channels of this analog device in this frame.
4 * Channel Count	Analog Data	32-bit float	Voltage samples with increasing channel order.

If the analog data has not been updated in QTM since last rt-packet, Analog Data will contain IEEE NaN (Not a number) float values.

Force components

There are two different force components that shares the same force header.

- Force
- Force single

BYTES	NAME	TYPE	DESCRIPTION

4	Component Size	32-bit integer	The size of the component including the header (Component Size, Component Type and Plate Count).
4	Component Type	32-bit integer	Value = 4 or 15. See Data component types.
4	Plate Count	32-bit integer	The number of force plates in this frame.

Repeated *Plate Count* times:

BYTES	NAME	TYPE	DESCRIPTION
4	Force Plate ID	32-bit integer	Id of the analog device in this frame. Id starts at 1.
4	Force Count	32-bit integer	The number of forces in this frame. **Only present for Analog component.** **Force Count is always 1 for Analog single component.**
4	Force Number	32-bit integer	Order number of first force in this frame. Force Number is increased with the force frequency. **Only present for Analog component.**
36 * Force Count	Force Data	32-bit float	X coordinate of the force Y coordinate of the force Z coordinate of the force X coordinate of the moment Y coordinate of the moment Z coordinate of the moment X coordinate of the force application point Y coordinate of the force application point Z coordinate of the force application point **If no force data is available for an Analog single component, Force Data will contain IEEE NaN (Not a number).**

Image component

BYTES	NAME	TYPE	DESCRIPTION
4	Component Size	32-bit integer	The size of the component including the header (Component Size, Component Type and Camera Count).
4	Component Type	32-bit integer	Value = 14. See Data component types.
4	Camera Count	32-bit integer	Number of cameras.

Repeated *Camera Count* times:

BYTES	NAME	TYPE	DESCRIPTION

4	Camera ID	32-bit integer	Camera ID of the camera which the image comes from. Id starts at 1.
4	Image Format	32-bit integer	Image format of the requested image. 0 = Raw Grayscale 1 = Raw BGR 2 = JPG 3 = PNG
4	Width	32-bit integer	Width of the requested image.
4	Height	32-bit integer	Height of the requested image.
4	Left Crop	32-bit float	Position of the requested image left edge relative the original image. 0: Original image left edge. 1: Original image right edge.
4	Top Crop	32-bit float	Position of the requested image top edge relative the original image. 0: Original image top edge. 1: Original image bottom edge.
4	Right Crop	32-bit float	Position of the requested image right edge relative the original image. 0: Original image left edge. 1: Original image right edge.
4	Bottom Crop	32-bit float	Position of the requested image bottom edge relative the original image. 0: Original image top edge. 1: Original image bottom edge.
4	Image Size	32-bit integer	Size of Image Data in number of bytes.
Image Size	Image Data	Binary data	Image data formatted according to the Image Format parameter.

Gaze vector

BYTES	NAME	TYPE	DESCRIPTION
4	Component Size	32-bit integer	The size of the component including the header (Component Size, Component Type and Gaze Vector Count).
4	Component Type	32-bit integer	Value = 16. See Data component types.
4	Gaze Vector Count	32-bit integer	Number of gaze vectors in this frame.

Repeated *Gaze Vector Count* times:

BYTES	NAME	TYPE	DESCRIPTION
4	Sample Count	32-bit integer	The size of the component including the header (Component Size, Component Type and Camera Count).
0 (Sample Count=0) 4 (Sample Count>0)	Sample Number	32-bit integer	Value = 16. See Data component types.
24 * Sample Count	Gaze Vector data	32-bit float	X component of the vector. Y component of the vector. Z component of the vector. X coordinate of the vector. Y coordinate of the vector. Z coordinate of the vector.

Eye Tracker

BYTES	NAME	TYPE	DESCRIPTION
4	Component Size	32-bit integer	The size of the component including the header (Component Size, Component Type and eye tracker Count).
4	Component Type	32-bit integer	Value = 19. See Data component types.
4	Eye Tracker Count	32-bit integer	Number of eye trackers in this frame.

Repeated *Eye Tracker Count* times:

BYTES	NAME	TYPE	DESCRIPTION
4	Sample Count	32-bit integer	The size of the component including the header (Component Size, Component Type and Camera Count).
0 (Sample Count=0) 4 (Sample Count>0)	Sample Number	32-bit float	Value = 19. See Data component types.
8 * Sample Count	Eye Tracker data	32-bit float	Left pupil diameter. Right pupil diameter.

Timecode

BYTES	NAME	TYPE	DESCRIPTION

4	Component Size	32-bit integer	The size of the component including the header (Component Size, Component Type and Timecode Count).
4	Component Type	32-bit integer	Value = 17. See Data component types.
4	Timecode Count	32-bit integer	Number of timecodes. For the time being QTM don't support multiple timecodes. **Timecode count is always 1**.

Repeated *Timecode Count* times:

BYTES	NAME	TYPE	DESCRIPTION
4	Timecode type	32-bit integer	The type of timecode. 0: SMPTE (32 bit) 1: IRIG (64 bit) 2: Camera time (64 bit)
4	Timecode Hi	32-bit integer	IRIG time code little endian format: Bit 0 – 6: Year Bit 7 – 15: Day of year Camera time Hi 32 bits of 64 bit integer timecode.
4	Timecode Lo	32-bit integer	IRIG time code little endian format: Bit 0 – 4: Hours Bit 5 – 10: Minutes Bit 11 - 16: Seconds bit 17 - 20: Tenth of a seconds SMPTE time code little endian format: Bit 0 – 4: Hours Bit 5 – 10: Minutes Bit 11 - 16: Seconds bit 17 - 21: Frame Bit 22 - 31 Not used Camera time Lo 32 bits of 64-bit integer timecode.

Skeleton

BYTES	NAME	TYPE	DESCRIPTION
4	Component Size	32-bit integer	The size of the component including the header (Component Size, Component Type and Skeleton Count).
4	Component Type	32-bit integer	Value = 18. See Data component types.
4	Skeleton	32-bit	Number of skeletons.

	Count	integer	

Repeated *Skeleton Count* times:

BYTES	NAME	TYPE	DESCRIPTION
4	Segment Count	32-bit integer	Number of segments in skeleton.

Repeated *Segment Count* times (32 * Segment Count Bytes):

BYTES	NAME	TYPE	DESCRIPTION
4	Segment ID	32-bit integer	ID of the segments in skeleton.
4	Position X	32-bit float	Segment position x coordinate.
4	Position Y	32-bit float	Segment position y coordinate.
4	Position Z	32-bit float	Segment position z coordinate.
4	Rotation X	32-bit float	Segment rotation quaternion x.
4	Rotation Y	32-bit float	Segment rotation quaternion y.
4	Rotation Z	32-bit float	Segment rotation quaternion z.
4	Rotation W	32-bit float	Segment rotation quaternion w.

If a skeleton is not visible in a frame, the segment count will be set to 0.

The rotation quaternion is sent in local coordinates. It can be changed to global coordinates by selecting skeleton:global as data type.

No More Data packet

This type of packet is sent when QTM is out of data to send because a measurement has stopped or has not even started.

BYTES	NAME	TYPE	VALUE
4	Size	32-bit integer	8 (only the header is sent).
4	Type	32-bit integer	4

C3D packet

This type of packet is sent as a response to the GetCaptureC3D command. It contains a C3D file, with the latest captured QTM measurement.

BYTES	NAME	TYPE	VALUE
4	Size	32-bit integer	8 + n (header bytes + C3D file size)

4	Type	32-bit integer	5
n	C3D file	Binary data	C3D file

QTM packet

This type of packet is sent as a response to the GetCaptureQTM command. It contains a C3D file, with the latest captured QTM measurement.

BYTES	NAME	TYPE	VALUE
4	Size	32-bit integer	8 + n (header bytes + C3D file size)
4	Type	32-bit integer	8
n	QTM file	Binary data	QTM file

Event packet

This type of packet is sent when QTM has an event to signal to the RT clients.

BYTES	NAME	TYPE	VALUE
4	Size	32-bit integer	9 (header bytes + event number)
4	Type	32-bit integer	6
1	Event	8-bit integer	Event number: 1-13, see Events.

Events

The RT server sends an event data packet to all its clients when the RT server changes state.

EVENT ID	NAME	COMMENT
1	Connected	Sent when QTM has connected to the camera system.
2	Connection Closed	Sent when QTM has disconnected from the camera system.
3	Capture Started	Sent when QTM has started a capture.
4	Capture Stopped	Sent when QTM has stopped a capture.
5	Not used	Previously Fetching Finished, deprecated.
6	Calibration Started	Sent when QTM has started a calibration.
7	Calibration Stopped	Sent when QTM has stopped a calibration.

8	RT From File Started	Sent when QTM has started real time transmissions from a file.
9	RT From File Stopped	Sent when QTM has stopped real time transmissions from a file.
10	Waiting For Trigger	Sent when QTM is waiting for the user to press the trigger button.
11	Camera Settings Changed	Sent when the settings have changed for one or more cameras. **Not included in the GetState response**.
12	QTM Shutting Down	Sent when QTM is shutting down. **Not included in the GetState response**.
13	Capture Saved	Sent when QTM has saved the current measurement. **Not included in the GetState response**.
14	Reprocessing Started	Sent when QTM has started reprocessing.
15	Reprocessing Stopped	Sent when QTM has stopped reprocessing.
16	Trigger	This event is sent by the server when QTM has received a trigger. **Not included in the GetState response**.

Discover packet

When this type of packet is broadcasted to QTM's auto discovery port, see IP port numbers, QTM responds with a discover response packet, see Discover response packet.

BYTES	NAME	TYPE	VALUE
4	Size	32-bit integer	10 (little endian)
4	Type	32-bit integer	7 (little endian)
2	Response Port	16-bit integer	Response port number: 0 - 65535. Network byte order (big endian).

Size and type is always sent as little endian 32 bit integers.

The Response Port is the UDP port number to which QTM sends a discover response message. The response port is big endian.

Discover response packet

The discover response packet is a special command message of type 1. The message contains a null terminated string, followed by the server's base port number.

BYTES	NAME	TYPE	VALUE
4	Size	32-bit integer	10 bytes. Little endian
4	Type	32-bit integer	1. Little endian
n+1	Server info string	Char	Null terminated string containing, server host name, QTM version and number of connected cameras. n = string size.
2	RT server base port.	16-bit integer	Base port number: 0 - 65535. Network byte order (Big endian).

Note: Size and Type is always sent as little endian 32 bit integers.

Example of a server info string: `MyComputer, QTM 2.5 (build 568), 5 cameras`.

Note: The base port number is only used for version 1.0 of the RT server, see IP port numbers to get the desired port number.

Open Sound Control (OSC).

The OSC version of the QTM RT server uses the Open Sound Control 1.0 specification.

Connecting (OSC)

When using the OSC protocol, which uses UDP, the client must first establish a connection with the server. This is because UDP is not connection-based like TCP. This is done with the `Connect` command, see Connect. A connection is closed with the disconnect command, see Disconnect.

The first thing that happens when you have connected to the QTM RT server with OSC is that the server sends a welcome message string: `QTM RT Interface connected`.

When using OSC it is not possible to set the protocol version, the latest version will always be used.

Port number (OSC)

There is only one server port available for OSC, base port + 4. OSC is sent via UDP packets. The clients listens to a UDP port for incoming OSC packets from the server. The client UDP server port is set to the RT server with the Connect command. See Connecting.

Commands (OSC)

In the description of the commands, number parameters are designated by an n, optional parameters are designated by enclosing brackets `[ ]` and choices between possible values are designated by a `|`. Parentheses are used to group parameters together. None of these characters (brackets, `|` or parentheses) should be included in the command sent to the server.

Command strings and their parameters never contain spaces, so a space character (ASCII 32) is used as separator between command names and parameters.

Command strings and parameter strings are case insensitive.

COMMAND	PARAMETERS
Connect	Port
Disconnect	
Version	
QTMVersion	
GetState	
GetParameters	`All` | `([General] [Calibration] [3D] [6D] [Analog] [Force] [Image] [GazeVector] [EyeTracker] [Skeleton])`
GetCurrentFrame	`[2D] [2DLin] [3D] [3DRes] [3DNoLabels] [3DNoLabelsRes] [Analog[:channels]] [AnalogSingle[:channels]] [Force] [ForceSingle] [6D] [6DRes] [6DEuler] [6DEulerRes] [Image] [GazeVector] [EyeTracker] [Timecode] [Skeleton[:global]]`
StreamFrames	`Stop` | `((FrequencyDivisor:n` | `Frequency:n` | `AllFrames) [UDP[:address]:port] ([2D] [2DLin] [3D] [3DRes][3DNoLabels] [3DNoLabelsRes] [Analog[:channels]] [AnalogSingle[:channels]] [Force] [ForceSingle] [6D] [6DRes] [6DEuler] [6DEulerRes] [Image] [GazeVector] [EyeTracker] [Timecode] [Skeleton[:global]]))`

Connect (OSC)

OSC Format:

```
Connect Port
```

Connects the client to the QTM RT server via the OSC protocol over UDP. The Port argument is the UDP port on which the client listens for server responses.

Disconnect (OSC)

Disconnects the client from the QTM RT server.

Version (OSC)

The server responds with `Version is n.n`, where `n.n` is the version of the RT protocol currently used.

It is not possible to set the version when connected via the OSC protocol. You can only retrieve current version.

Example:

```
Command:    Version
Response:   Version is 1.22
```

QTMVersion (OSC)

See standard version of the command, QTMVersion

GetState (OSC)

See standard version of the command,GetState

GetParameters (OSC)

See standard version of the command,GetParameters

GetCurrentFrame (OSC)

See standard version of the command, GetCurrentFrame

StreamFrames (OSC)

See standard version of the command, StreamFrames

QTM RT Packets (OSC)

Structure (OSC)

All OSC packets sent to or from the server have the same general layout. They don't have a header with size and type like the standard packet, see QTM RT Packets.

The content of the OSC packet differs slightly from the standard packet and uses the OSC data types for int32, int64, float32 and strings. All OSC packets sent to the RT server shall be sent in an OSC message with OSC address pattern `/qtm`. The address pattern of packets sent from the server depends on the packet type.

Packet types (OSC)

The Type field of a QTM RT server packet header is a number that should be interpreted according to the table below. These are the data types that are defined in the protocol so far. Detailed descriptions of the data packets for each type can be found in the sections following this one.

NAME	OSC ADDRESS	DESCRIPTION

Error	/qtm/error	The last command generated an error. The error message is included in the packet.
Command	/qtm	A command sent to the server.
Command response	/qtm/cmd_res	A response from the server to a command indicating that the command was successful.
XML	/qtm/xml	Data sent by the server in the form of XML, or data sent to the server in the form of XML.
Data frame header	/qtm/data	This message contains the data header and is followed by one or several data frame component messages, containing the real-time data sent from the server. The contents of the frame may vary depending on the commands/settings sent to the server. The contents may also vary between frames due to different sampling frequencies and buffering properties of different data types.
No More Data	/qtm/no_data	This packet type contains no data. It is a marker used to indicate that a measurement has finished or is not yet started.
Event	/qtm/event	This packet type contains event data from QTM.

Error packet (OSC)

Error packets are sent from the server only. Whenever you read a response from the server, it may be an error packet instead of the packet type you expect.

OSC error packets are sent in an OSC message with address pattern `/qtm/error` and contains one OSC string.

OSC TYPE	NAME	VALUE
OSC-string	Data	Example: "Command not supported."

Command packet (OSC)

OSC command packets sent to the RT server shall be sent in an OSC message with address pattern “/qtm”.

OSC TYPE	NAME	VALUE
OSC-string	Data	Example: "GetState"

Command response packet (OSC)

OSC command packets sent from the RT server as response to client commands is sent in a OSC message with address pattern `/qtm/cmd_res` .

OSC TYPE	NAME	VALUE
OSC-string	Data	"Connected"

XML packet (OSC)

The XML string contains the same data as for the standard XML Packet. OSC XML packets are sent in an OSC message with address pattern `/qtm/xml`.

OSC TYPE	NAME	VALUE
OSC-string	Data	XML string data. The XML data is described in XML Packet.

Data packet (OSC)

Each data frame is made up of one or more components, as specified in the commands GetCurrentFrame or StreamFrames. The data frame contains a Count that specifies the number of components in the frame. Every component starts with a component header – identical (in layout) to the packet header.

OSC data packets consist of one or several OSC messages enclosed in an OSC bundle. The first message contains the data frame header and has the OSC address pattern `/qtm/data`. It is followed by an OSC message for each data component. See OSC Data frame component types.

Data frame header (OSC)

The frame header and the data components are sent in an OSC bundle as separate OSC messages.

OSC TYPE	NAME	VALUE
Int32	Marker Timestamp Hi	Hi 32 bits of 64 bit timestamp value. Number of microseconds from start. The timestamp value is not valid for the Analog, Force and Gaze Vector data frame components, they have their own timestamps in their component data.
Int32	Marker Timestamp Lo	Lo 32 bits of 64 bit timestamp value. See above.
Int32	SMPTE TimeCode	SMPTE time code little endian format: Bit 0-4: Hours Bit 5-10: Minutes Bit 11-16: Seconds Bit 17-21: Frame Bit 22-30: Sub frame Bit 31: Valid bit
Int32	Marker Frame Number	The number of this frame. The frame number is not valid for the Analog, Force and Gaze Vector data frame components. They have their own sample numbers in their component data.
Int32	IRIG date	IRIG time code little endian format: Bit 0-6: Year Bit 7-15: Day

Int32	IRIG time	IRIG time code little endian format: Bit 0-4: Hours Bit 5-10: Minutes Bit 11-16: Seconds Bit 17-20: Tenth of a second
Int32	ComponentCount	The number of data components in the data message. Each component is sent as a separate OSC message.

Data frame component types (OSC)

Each data frame component has a unique OSC address. The table below shows the OSC address for all data components.

NAME	OSC ADDRESS	DESCRIPTION
2D	/qtm/2d	2D marker data
2D Linearized	/qtm/2d_lin	Linearized 2D marker data
3D	/qtm/3d	3D marker data. Each marker has its own OSC address. See OSC 3D component.
3D Residuals	/qtm/3d_res	3D marker data with residuals. Each marker has its own OSC address. See 3D with residuals component.
3D No Labels	/qtm/3d_no_labels	Unidentified 3D marker data.
3D No Labels Residuals	/qtm/3d_no_labels_res	Unidentified 3D marker data with residuals
Analog	/qtm/analog	Analog data from available devices.
Analog Single	/qtm/analog_single	Analog data from available analog devices. Only one sample per channel and camera frame. The latest sample is used if more than one sample is available.
Force	/qtm/force	Data from available force plates.
Force Single	/qtm/force_single	Force data from available force plates. Only one sample per plate and camera frame. The latest sample is used if more than one sample is available.
6D	/qtm/6d	6D data - position and rotation matrix. Each body has its own OSC address. See OSC 6DOF component.
6D Residuals	/qtm/6d_res	6D data - position and rotation matrix with residuals. Each body has its own OSC address. See 6DOF with residuals component.

6D Euler	/qtm/6d_euler	6D data - position and Euler angles. Each body has its own OSC address. See 6DOF Euler component.
6D Euler Residuals	/qtm/6d_euler_res	6D data - position and Euler angles with residuals. Each body has its own OSC address. See 6DOF Euler with residuals component.
Gaze Vector	/qtm/gaze_vector	Gaze vector data – Unit vector and position. Each gaze vector has its own OSC address.
Eye Tracker	/qtm/eye_tracker	Eye tracker data with pupil size.
Skeleton	/qtm/skeleton	Skeleton data – Position and rotation of all segments in the skeleton. Each skeleton has its own OSC address.

2D components (OSC)

There are two different 2D components.

- 2D
- 2D linearized

The 2D and 2D linearized data frame format are the same. The only difference is that the coordinates are linearized in 2D linearized.

OSC address: `/qtm/2d` or `/qtm/2d_lin` .

OSC TYPE	NAME	DESCRIPTION
Int32	Camera Count	Number of cameras. 32-bit integer.

Repeated *Camera Count* times:

OSC TYPE	NAME	DESCRIPTION
Int32	Marker Count	The number of markers for this camera in this frame.
	2D data	2D marker data from the camera, described below:

2D data, repeated *Marker Count* times:

OSC TYPE	NAME	DESCRIPTION
Int32	X	X coordinate of the marker.
Int32	Y	Y coordinate of the marker.
Int32	Diameter X	Marker X size.
Int32	Diameter Y	Marker Y size.

3D component (OSC)

Each marker is sent in a separate OSC message.

OSC address: `/qtm/3d/` The marker name is appended to the end of the address string. Example: `/qtm/3d/marker1` .

OSC TYPE	NAME	DESCRIPTION
Float32	X	X coordinate of the marker.
Float32	Y	Y coordinate of the marker.
Float32	Z	Z coordinate of the marker.

3D with residuals component (OSC)

Each marker is sent in a separate OSC message.

OSC address: `/qtm/3d_res/` The marker name is appended to the end of the address string. Example: `/qtm/3d_res/marker1` .

OSC TYPE	NAME	DESCRIPTION
Float32	X	X coordinate of the marker.
Float32	Y	Y coordinate of the marker.
Float32	Z	Z coordinate of the marker.
Float32	Residual	Residual for the 3D point.

3D no labels component (OSC)

OSC address: `/qtm/3d_no_labels/`

OSC TYPE	NAME	DESCRIPTION
Int32	Marker Count	The number of markers in this frame.

Repeated *Marker Count* times:

OSC TYPE	NAME	DESCRIPTION
Float32	X	X coordinate of the marker.
Float32	Y	Y coordinate of the marker.
Float32	Z	Z coordinate of the marker.
Int32	ID	An unsigned integer ID that serves to identify markers between frames.

3D no labels with residuals component (OSC)

OSC address: `/qtm/3d_no_labels_res/`

OSC TYPE	NAME	DESCRIPTION
Int32	Marker Count	The number of markers in this frame.

Repeated *Marker Count* times:

OSC TYPE	NAME	DESCRIPTION
Float32	X	X coordinate of the marker.
Float32	Y	Y coordinate of the marker.
Float32	Z	Z coordinate of the marker.
Int32	ID	An unsigned integer ID that serves to identify markers between frames.
Float32	Residual	Residual for the 3D point.

Analog component (OSC)

OSC address: `/qtm/analog/`

OSC TYPE	NAME	DESCRIPTION
Int32	Analog Device Count	Number of analog devices in this component.

Repeated *Analog Device Count* times:

OSC TYPE	NAME	DESCRIPTION
Int32	Analog Device ID	Id of this analog device.
Int32	Channel Count	The number of channels of this analog device in this frame.
Int32	Sample Count	The number of analog samples per channel in this frame.
Int32	Sample Number	Order number of first sample in this frame. Sample Number is increased with the analog frequency. There are Channel Count values per sample number.
Float32	Analog Data	There are (Channel Count * Sample Count) voltage values. The samples are ordered like this: Channel 1, Sample *Sample Number* Channel 1, Sample *Sample Number + 1* Channel 1, *Sample Sample Number + 2* .... Channel 1, Sample *Sample Number + Sample Count – 1* Channel 2, Sample *Sample Number* Channel 2, Sample *Sample Number + 1* ...

Analog single component (OSC)

OSC address: `/qtm/analog_single/

OSC TYPE	NAME	DESCRIPTION
Int32	Analog Device Count	Number of analog devices in this component.

Repeated *Analog Device Count* times:

OSC TYPE	NAME	DESCRIPTION
Int32	Analog Device ID	Id of this analog device.
Int32	Channel Count	The number of channels of this analog device in this frame.
Float32	Analog Data	There are Channel Count voltage values.

If there is no analog data available, Channel Count is set to 0 and Analog Data is omitted.

Force component (OSC)

OSC address: `/qtm/force/

OSC TYPE	NAME	DESCRIPTION
Int32	Plate Count	The number of force plates in this frame.

Repeated *Plate Count* times:

OSC TYPE	NAME	DESCRIPTION
Int32	Force Plate ID	Id of the analog device in this frame. Starts at 1.
Int32	Force Count	The number of forces in this frame.
Int32	Force Number	Order number of first force in this frame. Force Number is increased with the force frequency.
Float32	Force Data	There are *Force Count* force samples. Total size of the Force Data is 9 * *Force Count* Float32 values in following order: X coordinate of the force Y coordinate of the force Z coordinate of the force X coordinate of the moment Y coordinate of the moment Z coordinate of the moment X coordinate of the force application point Y coordinate of the force application point Z coordinate of the force application point

If Force Count = 0 (force not visible in QTM), Force Number and Force Data is omitted.

Force single component (OSC)

OSC address: `/qtm/force_single/

OSC TYPE	NAME	DESCRIPTION
Int32	Plate Count	The number of force plates in this frame.

Repeated *Plate Count* times:

OSC TYPE	NAME	DESCRIPTION
Int32	Force Plate ID	Id of the analog device in this frame. Starts at 1.
Float32	Force Data	Each force sample consists of 9 Float32 values in following order: X coordinate of the force Y coordinate of the force Z coordinate of the force X coordinate of the moment Y coordinate of the moment Z coordinate of the moment X coordinate of the force application point Y coordinate of the force application point Z coordinate of the force application point

If force not visible in QTM, Force Data is omitted.

6DOF component (OSC)

Each body is sent in a separate OSC message.

OSC address: `/qtm/6d/` The body name is appended to the end of the address string. Example: `/qtm/6d/body1` .

OSC TYPE	NAME	DESCRIPTION
Float32	X	X coordinate of the body.
Float32	Y	Y coordinate of the body.
Float32	Z	Z coordinate of the body.
Float32	Rotation	3x3 Rotation matrix of the body. Consists of 9 Float32 values.

6DOF with residuals component (OSC)

Each body is sent in a separate OSC message.

OSC address: `/qtm/6d_res/` The body name is appended to the end of the address string. Example: `/qtm/6d_res/body1` .

OSC TYPE	NAME	DESCRIPTION
Float32	X	X coordinate of the body.
Float32	Y	Y coordinate of the body.

Float32	Z	Z coordinate of the body.
Float32	Rotation	3x3 Rotation matrix of the body. Consists of 9 Float32 values.
Float32	Residual	Residual for the 6D body.

6DOF Euler component (OSC)

Each body is sent in a separate OSC message.

OSC address: `/qtm/6d_euler/` The body name is appended to the end of the address string. Example: `/qtm/6d_euler/body1` .

OSC TYPE	NAME	DESCRIPTION
Float32	X	X coordinate of the body.
Float32	Y	Y coordinate of the body.
Float32	Z	Z coordinate of the body.
Float32	Angle 1	First Euler angle, in degrees, as defined on the Euler tab in QTM's workspace options.
Float32	Angle 2	Second Euler angle.
Float32	Angle 3	Third Euler angle.

6DOF Euler with residuals component (OSC)

OSC address: `/qtm/6d_euler_res/` The body name is appended to the end of the address string. Example: `/qtm/6d_euler_res/body1` .

OSC TYPE	NAME	DESCRIPTION
Float32	X	X coordinate of the body.
Float32	Y	Y coordinate of the body.
Float32	Z	Z coordinate of the body.
Float32	Angle 1	First Euler angle, in degrees, as defined on the Euler tab in QTM's workspace options.
Float32	Angle 2	Second Euler angle.
Float32	Angle 3	Third Euler angle.
Float32	Residual	Residual for the 6D body.

Gaze vector component (OSC)

Each gaze vector is sent in a separate OSC message.

OSC address: `/qtm/gaze_vector/` The gaze vector name is appended to the end of the address string. Example: `/qtm/gaze_vector/Gaze Vector 1` .

OSC TYPE	NAME	DESCRIPTION
Int32	Sample Count	Number of vector samples in this frame.
Int32	Sample Number	Order number of first gaze vector sample in this frame. *Sample Number* is increased with the gaze vector frequency.

Repeated *Sample Count* times:

OSC TYPE	NAME	DESCRIPTION
Float32	Vector X	X component of the gaze unit vector.
Float32	Vector Y	Y component of the gaze unit vector.
Float32	Vector Z	Z component of the gaze unit vector.
Float32	Position Z	X coordinate of the gaze vector.
Float32	Rotation X	Y coordinate of the gaze vector.
Float32	Rotation Y	Z coordinate of the gaze vector.

Eye tracker component (OSC)

Each gaze vector is sent in a separate OSC message.

OSC address: `/qtm/eye_tracker/` The body name is appended to the end of the address string. Example: `/qtm/eye_tracker/Tobii` .

OSC TYPE	NAME	DESCRIPTION
Int32	Sample Count	Number of eye tracker samples in this frame.
Int32	Sample Number	Order number of first eye tracker sample in this frame. *Sample Number* is increased with the eye tracker frequency.

Repeated *Sample Count* times:

OSC TYPE	NAME	DESCRIPTION
Float32	Left pupil size	X component of the gaze unit vector.
Float32	Right pupil size	Y component of the gaze unit vector.

Skeleton component (OSC)

Each skeleton consists of several segments. All segments are sent in a separate OSC message.

OSC address: `/qtm/skeleton/` The skeleton and segment name is appended to the end of the address string. Example: `/qtm/skeleton/JohnDoe/Waist` .

OSC TYPE	NAME	DESCRIPTION
Int32	ID	Segment id.
Float32	Position X	Segment position x coordinate.
Float32	Position Y	Segment position y coordinate.
Float32	Position Z	Segment position z coordinate.
Float32	Rotation X	Segment rotation quaternion x.
Float32	Rotation Y	Segment rotation quaternion y.
Float32	Rotation Z	Segment rotation quaternion z.
Float32	Rotation W	Segment rotation quaternion w.

No More Data packet (OSC)

This type of packet is sent when QTM is out of data to send because a measurement has stopped or has not even started.

OSC no data packets are sent in a OSC message with address pattern `/qtm/no_data` .

OSC TYPE	NAME	VALUE
Nil	No data	OSC Nil type contains no data.

Event Data packet (OSC)

OSC event packets are sent in an OSC message with address pattern `/qtm/event` .

OSC TYPE	NAME	VALUE
OSC-string	Event	Event string. See OSC Events.

Events (OSC)

The RT server sends an event data packet to all its clients when the RT server changes state.

EVENT ID	NAME	COMMENT
1	Connected	Sent when QTM has connected to the camera system.
2	Connection Closed	Sent when QTM has disconnected from the camera system.
3	Capture Started	Sent when QTM has started a capture.
4	Capture Stopped	Sent when QTM has stopped a capture.

5	Fetching Finished	Sent when QTM has finished fetching a capture.
6	Calibration Started	Sent when QTM has started a calibration.
7	Calibration Stopped	Sent when QTM has stopped a calibration.
8	RT From File Started	Sent when QTM has started real time transmissions from a file.
9	RT From File Stopped	Sent when QTM has stopped real time transmissions from a file.
10	Waiting For Trigger	Sent when QTM is starting to wait for trigger to start a measurement.
11	Camera Settings Changed	Sent when the settings have changed for one or several cameras. Not included in the GetState command response.
12	QTM Shutting Down	Sent when QTM is shutting down. Not included in the GetState command response.
13	Capture Saved	Sent when QTM has saved current measurement. Not included in the GetState command response.
14	Reprocessing Started	Sent when QTM has started reprocessing.
15	Reprocessing Stopped	Sent when QTM has stopped reprocessing.

Telnet.

The OSC version of the QTM RT server uses the Open Sound Control 1.0 specification.

Connecting (Telnet)

Connect using the Telnet protocol on port 22221 of the QTM computer. Port 22221 (*base port – 1)* is the default Telnet port in QTM, see IP port numbers .

Commands (Telnet)

In the description of the commands, number parameters are designated by an *n*, optional parameters are designated by enclosing brackets [] and choices between possible values are designated by a '|'. Parentheses are used to group parameters together. None of these characters (brackets, '|' or parentheses) should be included in the command sent to the server.

Command strings and their parameters never contain spaces, so a space character (ASCII 32) is used as separator between command names and parameters.

Command strings and parameter strings are case insensitive.

COMMAND	**PARAMETERS**
Version	[n.n]
QTMVersion	
ByteOrder	
GetState	
GetParameters	`All | ([General] [Calibration] [3D] [6D] [Analog] [Force] [Image] [GazeVector] [EyeTracker] [Skeleton])`
StreamFrames	`Stop | ((FrequencyDivisor:n | Frequency:n | AllFrames) [UDP[:address]:port] ([2D] [2DLin] [3D] [3DRes][3DNoLabels] [3DNoLabelsRes] [Analog[:channels]] [AnalogSingle[:channels]] [Force] [ForceSingle] [6D] [6DRes] [6DEuler] [6DEulerRes] [Image] [GazeVector] [EyeTracker] [Timecode] [Skeleton[:global]]))`
TakeControl	`[Password]`
ReleaseControl	
New	
Close	-
Start	`[RTFromFile]`
Stop	-
Load	`Filename`
Save	`Filename [Overwrite]`
LoadProject	`ProjectPath`
Trig	
SetQTMEvent	`Label`
Reprocess	
Calibrate	[Refine]
Led	`Camera (On | Off | Pulsing) (Green | Amber | All)`
Quit	-

Version (Telnet)

```
Version
```

The server responds with *Version is n.n*, where *n.n* is the version of the RT protocol currently used.

It is not possible to set the version when connected via the Telnet protocol. You can only retrieve current version.

Example:

```
Command:    Version
Response:   Version is 1.22
```

QTMVersion (Telnet)

See standard version of the command, QTMVersion.

ByteOrder (Telnet)

See standard version of the command, ByteOrder.

GetState (Telnet)

```
GetState
```

This command makes the RT server send current QTM state as an event data packet. The event packet will only be sent to the client that sent the GetState command. `GetState` will not show the **Camera Settings Changed**, **QTM Shutting Down** and **Capture Saved events**.

Example:

```
Command:    GetState

Response:   Connected                 or
            Connection Closed         or
            Capture Started           or
            Capture Stopped           or
            Capture Fetching Finished or
            Calibration Started       or
            Calibration Stopped       or
            RT From File Started      or
            RT From File Stopped      or
            Waiting For Trigger       or
            Reprocessing Started      or
            Reprocessing Stopped
```

GetParameters (Telnet)

See standard version of the command, GetParameters.

StreamFrames (Telnet)

See standard version of the command, StreamFrames.

TakeControl (Telnet)

See standard version of the command, TakeControl.

ReleaseControl (Telnet)

See standard version of the command, ReleaseControl.

New (Telnet)

See standard version of the command, New.

Close (Telnet)

See standard version of the command, Close.

Start (Telnet)

See standard version of the command, Start.

Stop (Telnet)

See standard version of the command, Stop.

Load (Telnet)

See standard version of the command, Load.

Save (Telnet)

See standard version of the command, Save.

LoadProject (Telnet)

See standard version of the command, LoadProject.

Trig (Telnet)

See standard version of the command, Trig.

SetQTMEvent (Telnet)

See standard version of the command, SetQTMEvent.

Reprocess (Telnet)

See standard version of the command, Reprocess.

Calibrate (Telnet)

See standard version of the command, Calibrate.

Led (Telnet)

See standard version of the command, Led.

Quit (Telnet)

See standard version of the command, Quit.

Changelog.

Changes in 1.22

- Moved solver from skeleton to segment and added constraint, couplings and goal to DegreesOfFreedom in skeleton XML settings.

Changes in 1.21

- Changed skeleton xml format and allow sending skeleton settings to QTM.
- Changed 6d xml format and allow sending 6d settings to QTM.
- Added `EyeTracker` component with pupil diameter.

Changes in 1.20

- Added new settings for rigid body points: `PhysicalId` and `Virtual` .
- Removed `Camera_System` from settings.
- Only allow save command to save measurement if QTM is not connected to the camera system.
- Added `Calibrate` command and calibration settings.

Changes in 1.19

- Added new data component, `Skeleton` .
- Removed data component `All` from `GetCurrentFrame` and `StreamFrames` .

Changes in 1.18

- Added `Miqus Video Color` camera type.
- Added Auto white balance camera settings

Changes in 1.17

- Added support for external time base `IRIG` .
- Added `IRIG` time code to OSC frame header.
- Added new data component, `Timecode` .
- Added new event type, `Trigger` .
- Added Auto exposure camera settings.

Changes in 1.16

- Added `Miqus Video` camera type.
- Removed Video modes settings from general camera settings.
- Added video Resolution and Aspect_Ratio settings to general camera settings.
- Added Euler rotation names to 6DOF settings.
- Added Lens Control focus and aperture settings.

Changes in 1.15

- Added `Led` command.
- Added `Reprocess` command.
- Added `Miqus Sync Unit` camera type.
- Added general camera settings for Miqus Sync Unit trigger settings (`Start_On_Trigger_NO` , `Start_On_Trigger_NC` , `Start_On_Trigger_Software`)
- Added general camera setting, `Supports_HW_Sync` , `Sync_Out2` and `Sync_Out_MT` .
- Removed SRAM wired sync out mode.
- Added `Camera_System` and subvalue `Type` to general XML.

Changes in 1.14

- Added bone color to 3d XML parameters.

- Added support for new processing action: `PreProcessing2D` .
- Added support for real-time processing actions and reprocessing actions settings.
- Changed XML settings tag from `Duty cycle` to `Duty_Cycle` .
- Added option to only stream data from selected analog channels.

Changes in 1.13

- Added export to AVI file and gaze vector processing actions.
- Updated Telnet protocol version.
- Made it possible to change video mode and video capture frequency.
- Changes to force calibration matrix. Now supports more than 6x6 matrixes.
- Added support for trajectory bones.

Changes in 1.12

- Added `Load` function for loading measurements in QTM.
- Added `LoadProject` function for loading project in QTM.
- Added new sync out mode SRAM wired in General/Camera settings.

Changes in 1.11

- Changed analog XML parameters. Now all channels have their own unit setting.
- Changed timestamp in OSC data frame header, from one 64 bit integers, to two 32 bit integers (hi and lo word).

Changes in 1.10

- Added general camera XML parameters `External_Time_Base` ,
- `Processing_Actions` and Camera/Underwater.
- Added 3D XML parameters `CalibrationTime` .
- Changed Save command. Added overwrite parameter.
- Made it possible to change the capture frequency via the frequency general setting.
- Changed `GetLastEvent` to `GetState` .
- Support fetching of General and Image parameters even if QTM is not connected to a camera system.
- Changed the Close command. It will now respond with `Closing file` instead of `Closing connection` when not in RT (preview) mode. The “No connection to close” response is now sent as a command packet, not an error packet.

- Changed `New` command response. The `Already connected` response is now sent as a command packet, not an error packet.
- Added Capture Saved event.
- Removed `Fetching Finished` event.
- Added `GetCaptureQTM` command.
- Changed `GetCapture` command response. Send a command packet with `Sending capture` before sending the XML packet with the capture file.
- Added RT server base port to discover response packet.

Changes in 1.9

- Added `ForceSingle` data component.
- Fixed bug in OSC `Analog` , `AnalogSingle` and `Force` data components.
- Fixed bug in OSC `3DNoLabels` data component.
- Allow capture start via RT server even if camera system isn't calibrated.

Changes in 1.8

- Added events: `Camera Settings Changed` and `QTM Shutting Down` .
- Added RT server auto discovery.
- New data frame component: `Image`
- Added new XML setting: Image and General Camera setting Orientation.
- GetParameters command returns `Parameters not available` error string, instead of a `No More Data” package` .
- Added status byte to `2D` and `2DLin` data components.
- Changed all 64-bit float coordinates to 32-bit floats in the 3D and 6DOF data frames.
- Removed all 32-bit padding form the protocol.
- Don't broadcast string `Server shutting down` to all clients when shutting down. Use event `QTM Shutting Down` instead.
- Added password to `TakeControl` .
- Fixed bug in `AnalogSingle` Big Endian data component.
- Changed name of `GetCapture` to `GetCaptureC3D`
- Changed Force plate identification in XML strings from `Force_Plate_Index` to `Plate_ID` .
- Changed name of Event command to `SetQTMEvent` .

Changes in 1.7

- Added `Trig` command.

- Added `Event` command.
- Added event: `Waiting For Trigger` .
- Changed format of XML data packet and added new XML setting. General setting: `Start On External Trigger` .

Changes in 1.6

- Added OSC support.
- Apply rotation and translation to 6 DOF bodies.
- Added `Camera` to general XML parameters.
- Added `Save` command.

Changes in 1.5

- Added new command: `QTMVersion` .
- `Version` command without argument will return current version used by the server.
- Added new general parameter: `Capture Time` .
- Added possibility to change settings via an XML parameter file. Supported settings: `Capture time` and Force plate corners.
- Added new commands: `New` , `Close` , `Start` , `Stop` , `GetCapture` and `GetLastEvent` .
- Added events: `Connected` , `Closed` , `Capture started` and `Capture stopped` .

Changes in 1.4

- Added 6D (6 DOF) XML parameters.
- Added color to 3D XML parameters.
- Removed `LicenceName` argument in the `CheckLicense` command.

Changes in 1.3

- Added `2D Drop Rate` and `2D Out Of Sync Rate` to frame component header for: 3d, 3DRes, 3DnoLabels and 3DNoLabelsRes.

Changes in 1.2

- `2DLin` , `3DRes` , `3DNoLabelsRes` , `6DRes` and `6DEulerRes` data type components were added.

- CheckLicense command added.
- `<AxisUpwards>` item added to XML parameters for 3D.

Changes in 1.1

- UDP support added to the `StreamFrames` command.
- Analog data frame component changed. Includes sample number and can contain several samples per channel in a single data frame.
- Force data frame component changed. Includes sample number and can contain several samples per force plate in a single data frame.
- Analog parameters changed, device ID added.
- Force parameters changed, device ID added.
- `SendParameters` command changed to `GetParameters` .