Written by	Document number	Revision	Date
Qualisys Software Team	QDEV-QTM-EXTERN-SDK_RT-QTM_RT_SERVER_V_1_15	1.0	2016-08-24

QTM Real-time Server Protocol Documentation

Version 1.15

Document number	Title	Revision	Date
QDEV-QTM-EXTERN-SDK_RT-QTM_RT_SERVER_V_1_15	QTM RT server protocol documentation - v 1.15	1.0	2016-08-24

Contents

1 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.

2 PROTOCOL VERSIONS

This document describes the 1.15 version of the QTM RT server protocol.

2.1 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 (in section 5.10.1 on page 89), there is a list of the changes that have been made to the protocol between different versions.

2.2 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 QTM RT server using OSC, differs from the standard version of the RT protocol. See 5.1.

2.3 Telnet

The Telnet protocol will always connect to QTM using the latest version of the protocol. Connecting to the QTM RT server using Telnet, differs from the standard version of the RT protocol. See 6.1.

3 OVERVIEW

3.1 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 3.2.2. The *discover packet* shall contain the port number to which QTM sends an UDP response string, see 4.34. Except for the IP address, the client will also respond with the host name, QTM version and number of connected cameras.

3.2 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.

WARNING: By not setting protocol version, QTM will default to version 1.1. This will lead to very strange behaviour since the frame sizes and data formats have changed a lot since 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 3.2.1.

3.2.1 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.

3.2.2 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). It 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.

3.3 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 (see 4.33 and 5.10) or an error (see 4.26 and 5.4) message could arrive when a completely different response is expected.

3.4 Retrieving settings

Before requesting streamed data, it may be necessary to ask QTM about different settings, for example what frequency the system is capturing in and what labels the labeled markers have. For all such information that does not change with each frame, the command *GetParameters* is used. QTM replies with an XML data string package, that the client should parse and extract the required data from. See 4.28.

3.5 Change settings

It is possible to change some of the QTM settings via the RT server. This is done by sending an XML data packet, see 4.28, containing the settings to be changed. Settings that are possible to change are: *General*, *Image* and *Force*, See 3.5.1, 3.5.2 and 3.5.3.

If the settings were updated ok, the server will send a “Setting parameters succeeded” command string response. Otherwise a “Setting parameters failed” error string will be sent.

Change settings is not available with the OSC protocol.

3.5.1 General settings

Frequency

The Frequency setting tells QTM at which frequency the cameras shall operate. The frequency is expressed in Hz.

Capture_Time

The Capture_Time setting tells QTM how long a capture started with the *start* command shall be. The time is expressed in seconds.

Start_On_External_Trigger

The Start_On_External_Trigger setting tells QTM if the measurement shall start on external trigger. The value can be true or false. Legacy parameter that for oqus is the trigger input but for the Miqus Sync Unit specifies any of the Trig NO or Trig NC ports.

Start_On_Trigger_NO

The Start_On_Trigger_NO setting tells QTM if the measurement shall start on external trigger signal from Miqus Sync Unit Trig NO port or the Oqus trigger input. The value can be true or false.

Start_On_Trigger_NC

The Start_On_Trigger_NC setting tells QTM if the measurement shall start on external trigger signal from Miqus Sync Unit Trig NC port. The value can be true or false.

Start_On_Trigger_Software

The Start_On_Trigger_Software setting tells QTM if the measurement shall start on external trigger signal from a software trigger. It can be devices and applications like keyboard, RT clients, telnet command etc. The value can be true or false.

External_Time_Base

Enabled

Enable or disable external time base. Value can be True or False.

Signal_Source

Signal source used for external time base. Selectable values:

- Control port
- IR receiver
- SMPTE
- Video sync

Signal_Mode

Selectable values:

- Periodic
- Non-periodic

Frequency_Multiplier

Multiply incoming frequency by this integer to get the camera frequency. Can be combined with frequency divisor. Value is an integer.

Frequency_Divisor

Divide incoming frequency by this integer to get the camera frequency. Can be combined with frequency multiplier. Value is an integer.

Frequency_Tolerance

Frequency tolerance in ppm of period time. Value is an integer.

Nominal_Frequency

Nominal frequency used by QTM. To disable nominal frequency set the value to *None*. To enable nominal frequency set a float value.

Signal_Edge

Control port TTL signal edge.

- Negative
- Positive

Signal_Shutter_Delay

Delay from signal to shutter opening in micro seconds. Value is an integer.

Non_Periodic_Timeout

Max number of seconds expected between two frames in non-periodic mode. Value is a float.

Processing_Actions

PreProcessing2D

Enable or disable 2D pre-processing action. Value can be True or False.

Tracking

Enable or disable 3D or 2D tracking processing action. Value can be 3D, 2D or False.

TwinSystemMerge

Enable or disable twin system merge processing action. Value can be True or False.

SplineFill

Enable or disable spline fill processing action. Value can be True or False.

AIM

Enable or disable AIM processing action. Value can be True or False.

Track6DOF

Enable or disable 6DOF tracking processing action. Value can be True or False.

ForceData

Enable or disable force data processing action. Value can be True or False.

GazeVector

Enable or disable gaze vector data processing action. Value can be True or False.

ExportTSV

Enable or disable export to TSV processing action. Value can be True or False.

ExportC3D

Enable or disable export to C3D processing action. Value can be True or False.

ExportMatlabFile

Enable or disable export to MATLAB file processing action. Value can be True or False.

ExportAviFile

Enable or disable export to AVI video file processing action. Value can be True or False.

RealTime_Processing_Actions

PreProcessing2D

Enable or disable 2D pre-processing action. Value can be True or False.

Tracking

Enable or disable 3D tracking processing action. Value can be 3D or False.

AIM

Enable or disable AIM processing action. Value can be True or False.

Track6DOF

Enable or disable 6DOF tracking processing action. Value can be True or False.

ForceData

Enable or disable force data processing action. Value can be True or False.

GazeVector

Enable or disable gaze vector data processing action. Value can be True or False.

ExportTSV

Enable or disable export to TSV processing action. Value can be True or False.

ExportC3D

Enable or disable export to C3D processing action. Value can be True or False.

ExportMatlabFile

Enable or disable export to MATLAB file processing action. Value can be True or False.

ExportAviFile

Enable or disable export to AVI video file processing action. Value can be True or False.

Reprocessing_Actions

PreProcessing2D

Enable or disable 2D pre-processing action. Value can be True or False.

Tracking

Enable or disable 3D or 2D tracking processing action. Value can be 3D, 2D or False.

TwinSystemMerge

Enable or disable twin system merge processing action. Value can be True or False.

SplineFill

Enable or disable spline fill processing action. Value can be True or False.

AIM

Enable or disable AIM processing action. Value can be True or False.

Track6DOF

Enable or disable 6DOF tracking processing action. Value can be True or False.

ForceData

Enable or disable force data processing action. Value can be True or False.

GazeVector

Enable or disable gaze vector data processing action. Value can be True or False.

ExportTSV

Enable or disable export to TSV processing action. Value can be True or False.

ExportC3D

Enable or disable export to C3D processing action. Value can be True or False.

ExportMatlabFile

Enable or disable export to MATLAB file processing action. Value can be True or False.

ExportAviFile

Enable or disable export to AVI video file processing action. Value can be True or False.

Camera

General settings consist of none or several *Camera* elements, with following content.

ID

Select camera to which the settings shall apply. If the camera id is set to a negative value, settings will apply to all cameras. This value must always be present.

Mode

Changes camera mode for selected camera. Available camera modes are:

- Marker
- Marker Intensity
- Video

Video_Mode

Set video mode. This setting only apply to specific camera models that are not used for marker detection. Valid settings are:

- Custom
- 1080p_24hz
- 720p_25hz
- 720p_50hz
- 540p_25hz
- 540p_50hz
- 540p_60hz
- 480_25hz
- 480_50hz
- 480_60hz
- 480_120hz

Video_Frequency

Set video capture frequency for the camera selected by Camera ID, see above. The value is either in Hz (> 1 Hz) or in percent of max frequency (0.0 to 1.0), 32-bit float. Note: It is only possible to set minimum video capture frequency, which is 1 Hz, by setting the Video_Frequency setting to 0 (0 %).

Video_Exposure

Set video exposure time for the camera selected by Camera ID, see above. The value is either in micro seconds (> 5 µs) or in percent of max value (0.0 to 1.0), 32-bit float.

Video_Flash_Time

Set video flash time for the camera selected by Camera ID, see above. The value is either in micro seconds (> 5 µs) or in percent of max value (0.0 to 1.0), 32-bit float.

Marker_Exposure

Set marker exposure time for the camera selected by Camera ID, see above. The value is either in micro seconds (> 5 µs) or in percent of max value (0.0 to 1.0), 32-bit float.

Marker_Threshold

Set marker threshold for the camera selected by Camera ID, see above. The value is either an absolute value (50 – 900) or in percent of max value (0.0 to 1.0), 32-bit float.

Orientation

Set camera orientation for the camera selected by Camera ID, see above. The setting affects the 2D camera view in QTM. The value is in degrees (0, 90, 180 or 270), 32-bit integer.

Sync_Out/Sync_Out2

Camera settings consist of none, one *Sync_Out*, or one *Sync*_Out2 block, with following content:

Mode

Synchronization mode for the selected camera. Available modes:

- Shutter out
- Multiplier
- Divisor
- Camera independent
- Measurement time
- Continuous 100Hz

Value

This integer value is only used for three of the sync out modes. The content is different depending on the *Mode* setting.

- Multiplier — Multiplier applied to the camera frequency
- Divisor — Divisor applied to the camera frequency
- Camera independent — Camera independent frequency

Duty_Cycle

Output duty cycle in per cent (float). Only used in multiplier, divisor and camera independent mode.

Signal_Polarity

TTL signal polarity. Possible values:

- Positive
- Negative

Sync_Out_MT

Camera settings consist of none or one *Sync_Out_MT* block, with following content:

Signal_Polarity

TTL signal polarity. Possible values:

- Positive
- Negative

3.5.2 Image settings

The *Image* element in the XML data packet consists of none or several *Camera* elements. The image settings are used to request streaming images from one or several cameras.

Camera

The settings within a *Camera* element must come in a predefined order, see below and example 3.5.4. All settings can be set individually, except for ID, which always has to be present. If the selected camera is not enabled since before, the default values will be used for all image settings that are not present in the *Camera* element. Otherwise current image settings will be used.

ID

Select camera to fetch images from. This value must always be present in the image settings.

Enabled

Enable or disable transmission of image data from camera selected by Camera ID, see above.

True or False

Format

Available image formats.

RAWGrayscale

RAWBGR
JPG (Default)
PNG

Width

Width of the requested image. This does not take into account the cropping. The width is the dimensions had the image not been cropped at all. 32-bit integer.

Height

Height of the requested image. This does not take into account the cropping. The height is the dimensions had the image not been cropped at all. 32-bit integer.

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 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 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 requested image bottom edge relative the original image. 32-bit float.
0.0 = Original image top edge.
1.0 = Original image bottom edge (Default).

3.5.3 Force settings

The Force section in the XML data packet consists of none or several *Plate* elements.

Plate

Each *Plate* element consists of a *Force_Plate_Index* and a *Location* element. The settings within a plate element must come in a predefined order, see example 3.5.4.

Force_ID

ID of camera to fetch images from. This value must always be present in the image settings.

Location

The *Location* element consists of four corner elements: *Corner1*, *Corner2*, *Corner3* and *Corner4*. Each corner element consists of X, Y and Z elements with the coordinates for the force plate (32 bit floats).

3.5.4 Settings example

Send the following XML data packet to the RT server:

```
<QTM_Settings>
    <General>
        <Capture_Time>2.5</Capture_Time>
        <Capture_Frequency>25</Capture_Frequency>
        <Start_On_External_Trigger>True</Start_On_External_Trigger>
        <Start_On_Trigger_NO>True</Start_On_Trigger_NO>
        <Start_On_Trigger_NC>False</Start_On_Trigger_NC>
        <Start_On_Trigger_Software>False</Start_On_Trigger_Software>
        <External_Time_Base>
            <Enabled>True</Enabled>
            <Signal_Source>Control port</Signal_Source>
            <Signal_Mode>Periodic</Signal_Mode>
            <Frequency_Multiplier>1</Frequency_Multiplier>
            <Frequency_Divisor>1</Frequency_Divisor>
            <Frequency_Tolerance>1000</Frequency_Tolerance>
            <Nominal_Frequency>None</Nominal_Frequency>
            <Signal_Edge>Negative</Signal_Edge>
            <Signal_Shutter_Delay>10000</Signal_Shutter_Delay>
            <Non_Periodic_Timeout>10</Non_Periodic_Timeout>
        </External_Time_Base>
        <Processing_Actions>
            <PreProcess2D>False</PreProcess2D>
            <Tracking>3D</Tracking>
            <TwinSystemMerge>False</TwinSystemMerge>
            <SplineFill>True</SplineFill>
            <AIM>True</AIM>
            <Track6DOF>False</Track6DOF>
            <ForceData>False</ForceData>
            <GazeVectorData>False</GazeVectorData>
            <ExportTSV>False</ExportTSV>
            <ExportC3D>False</ExportC3D>
            <ExportMatlabFile>False</ExportMatlabFile>
        </Processing_Actions>
        <RealTime_Processing_Actions>
            <PreProcess2D>False</PreProcess2D>
            <Tracking>3D</Tracking>
            <AIM>True</AIM>
            <Track6DOF>False</Track6DOF>
            <ForceData>False</ForceData>
            <GazeVectorData>False</GazeVectorData>
        </RealTime_Processing_Actions>
        <Reprocessing_Actions>
            <PreProcess2D>False</PreProcess2D>
            <Tracking>3D</Tracking>
            <TwinSystemMerge>False</TwinSystemMerge>
```

```
            <SplineFill>True</SplineFill>
            <AIM>True</AIM>
            <Track6DOF>False</Track6DOF>
            <ForceData>False</ForceData>
            <GazeVectorData>False</GazeVectorData>
            <ExportTSV>False</ExportTSV>
            <ExportC3D>False</ExportC3D>
            <ExportMatlabFile>False</ExportMatlabFile>
        </Reprocessing_Actions>
        <Camera>
            <ID>1</ID>
            <Mode>Marker</Mode>
            <Video_Mode>Custom</Video_Mode>
            <Video_Frequency>30</Video_Frequency>
            <Video_Exposure>0.5</Video_Exposure>
            <Video_Flash_Time>0.3</Video_Flash_Time>
            <Marker_Exposure>0.5</Marker_Exposure>
            <Marker_Threshold>0.4</Marker_Threshold>
            <Orientation>0</Orientation>
            <Sync_Out>
                <Mode>Camera independent</Mode>
                <Value>120</Value>
                <Duty_Cycle>50.000</Duty_Cycle>
                <Signal_Polarity>Negative</Signal_Polarity>
            </Sync_Out>
        </Camera>
    </General>
    <Image>
        <Camera>
            <ID>1</ID>
            <Enabled>True</Enabled>
            <Format>JPG</Format>
            <Width>640</Width>
            <Height>400</Height>
            <Left_Crop>0.0</Left_Crop>
            <Top_Crop>0.0</Top_Crop>
            <Right_Crop>1.0</Right_Crop>
            <Bottom_Crop>1.0</Bottom_Crop>
        </Camera>
    </Image>
    <Force>
        <Plate>
            <Force_ID>1</Force_ID>
            <Location>
                <Corner1>
                    <X>1000.00</X>
                    <Y>0.00</Y>
                    <Z>0.00</Z>
                <Corner1>
                <Corner2>
                    <X>1600.00</X>
                    <Y>0.00</Y>
                    <Z>0.00</Z>
                <Corner2>
                <Corner3>
                    <X>1600.00</X>
                    <Y>400.00</Y>
```

```
                    <Z>0.00</Z>
                <Corner3>
                <Corner4>
                    <X>1000.00</X>
                    <Y>400.00</Y>
                    <Z>0.00</Z>
                <Corner4>
            </Location>
        </Plate>
    </Force>
</QTM_Settings>
```

Response: *Setting parameters succeeded* or
Setting parameters failed

3.6 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.

4 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 (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.

The response to a command is a command packet, error packet, XML packet, C3D data packet or QTM data packet. Each command bellow 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”.

Command summary

Version	[n.n]
QTMVersion	
ByteOrder	
GetState	
GetParameters	All | ([General] [3D] [6D] [Analog] [Force] [Image] [GazeVector])
GetCurrentFrame	All | ([2D] [2DLin] [3D] [3DRes] [3DNoLabels] [3DNoLabelsRes] [Analog] [AnalogSingle] [Force] [ForceSingle] [6D] [6DRes] [6DEuler] [6DEulerRes] [Image] [GazeVector])
StreamFrames	Stop | ((FrequencyDivisor:n | Frequency:n | AllFrames) [UDP[:address]:port] (All | ([2D] [2DLin] [3D] [3DRes] [3DNoLabels] [3DNoLabelsRes] [Analog] [AnalogSingle] [Force] [ForceSingle] [6D] [6DRes] [6DEuler] [6DEulerRes] [Image] [GazeVector])))
TakeControl	[Password]
ReleaseControl	
New	
Close	
Start	[RTFromFile]
Stop	
Load	Filename
Save	Filename [Overwrite]
LoadProject	ProjectPath
GetCaptureC3D	
GetCaptureQTM	
Trig	
SetQTMEvent	Label
Quit	
Led	(camera) (On | Off | Pulsing) (Green | Amber | All)

4.1 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. It is not possible to change version while streaming data.

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

WARNING: By not setting protocol version, QTM will default to version 1.1. This will lead to very strange behaviour since the frame sizes and data formats have changed a lot since version 1.1.

Example: Command: *Version 1.15*
Response: *Version set to 1.15* or
Version NOT supported *or*
Cannot change version while streaming data

Command: *Version*
Response: *Version is 1.15*

4.2 QTMVersion

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

Example: Command: *QTMVersion*
Response: *QTM Version is 2.3 (build 464)*

4.3 ByteOrder

Returns the current byte order.

Example: Command: *ByteOrder*
Response: *Byte order is little endian* or
Byte order is big endian

4.4 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.

Example: Command: *GetState*
Response: Event data packet with last QTM event.

Response (Telnet): *Connected* or
Connection Closed or
Capture Started or
Capture Stopped or
Calibration Started or
Calibration Stopped or
RT From File Started or
RT From File Stopped or
Waiting For Trigger

4.5 GetParameters

GetParameters All | ([General] [3D] [6D] [Analog] [Force] [Image] [GazeVector])

This command retrieves the settings for the requested component(s) of QTM in XML format. The XML parameters are described in section 4.28 on page 34.

Example: Command: *GetParameters 3D Force*
Response: *Parameters not available* or
XML string containing requested parameters

4.6 GetCurrentFrame

GetCurrentFrame All |
([2D] [2DLin] [3D] [3DRes] [3DNoLabels] [3DNoLabelsRes]
[Analog[:*channels*]] [AnalogSingle[:*channels*]]
[Force] [ForceSingle] [6D] [6DRes] [6DEuler] [6DEulerRes]
[GazeVector] [Image])

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

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 section 4.29 on page 55.
- 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 (see section 4.30 on page 69).

- 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.

Example: Command: *GetCurrentFrame 3D Analog*
Response: One data frame is sent to the client according to the data frame protocol described in section 4.29 on page 55.

4.7 StreamFrames

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

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

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 section 4.29 on page 55.
- 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 (see section 4.30 on page 69).
- 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 a rate of approximately n Hz.
    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 60Hz. This means that usually every other processed frame is transmitted, but once in a while two frames in a row are transmitted (to reach 60Hz instead of 50).
  - **AllFrames**
    When a client specifies *AllFrames* in the *StreamFrames* command, every real-time frame processed by QTM is transmitted to the 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 (see section 4.30 on page 69).

- 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.

Example: Command: *StreamFrames Frequency:30 UDP:2234 3D Analog*
Response: 30 frames per second containing 3D data and Analog data are streamed over UDP/IP to the client computer's port 2234. The data frame protocol is described in section 4.29 on page 55.

4.8 TakeControl

`TakeControl` `[Password]`

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. 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.

Example: Command: *TakeControl x364k6Gt*
Response: *You are now master* or
You are already master or
127.0.0.1 (1832) is already master or
Client control disabled in QTM *or*
Wrong or missing password

4.9 ReleaseControl

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

Example: Command: *ReleaseControl*
Response: *You are now a regular client* or
You are already a regular client

4.10 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 4.8.

Example:	Command:	*New*	
	Response:	*Creating new connection*	or
		Already connected	or
		The previous measurement has not been saved or closed	or
		You must be master to issue this command	

4.11 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 4.8.

Example:	Command:	*Close*	
	Response:	*Closing connection*	or
		Closing file	or
		No connection to close	or
		You must be master to issue this command	

4.12 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 4.8.

Example:	Command:	*Start*	
	Response:	*Starting measurement*	or
		Measurement is already running	or
		Not connected. Create connection with new	or
		Starting RT from file	or
		RT from file already running	or
		No file open	or
		You must be master to issue this command	

4.13 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 4.8.

Example:	Command:	*Stop*	
	Response:	*Stopping measurement*	or
		No measurement is running	or
		Parse error	or
		You must be master to issue this command	

4.14 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 4.8.

Example:	Command:	*Load*	
	Response:	*Measurement loaded*	or
		Missing file name	or
		Failed to load measurement	or
		Active camera connection exists	or
		Current measurement not saved	or
		Parse error	or
		You must be master to issue this command	

4.15 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 (_##).

It is only possible to issue this command if you have the control over the QTM RT interface. See 4.8.

Example: Command: *Save*

Response:	*Measurement saved*	or
	Measurement saved as 'new filename with counter'	or
	Failed to save measurement	or
	No write access.	or
	Failed to create directory	or
	No measurement to save	or
	You must be master to issue this command	

4.16 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. C:\Users\lnn\QTM files\Imagination

It is only possible to issue this command if you have the control over the QTM RT interface. See 4.8.

Example: Command: *Load*

Response:	*Project loaded*	or
	Missing project name	or
	Failed to load project	or
	Active camera connection exists	or
	Current measurement not saved	or
	Parse error	or
	You must be master to issue this command	

4.17 GetCaptureC3D

This command will download the latest capture as a C3D file.

Example: Command: *GetCapture*
Response: *Sending capture* followed by a C3D data packet containing current capture. or
No capture to get or
Error sending C3D file

4.18 GetCaptureQTM

This command will download the latest capture as a QTM file.

Example: Command: *GetCapture*
Response: *Sending capture* followed by a QTM data packet containing current capture. or
No capture to get or
Error sending QTM file

4.19 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 4.33.1. It is only possible to issue this command if you have the control over the QTM RT interface. See 4.8.

Example: Command: *Trig*
Response: *Trig ok* or
QTM not waiting for trig or
You must be master to issue this command

4.20 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”. It is only possible to issue this command if you have the control over the QTM RT interface. See 4.8.

This command will set an event in QTM.

Example: Command: *Event test_event*
Response: *Event set* or
Event label too long or
QTM is not capturing or
You must be master to issue this command

4.21 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 4.8.

Example: Command: Reprocess
Response: *Reprocessing file* or
No file open or
RT from file running

4.22 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.

Example: Command: *Quit*
Response: *Bye bye*

4.23 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).

Example: Command: *Led*
Response: *Parse error* or
You must be master to issue this command

Protocol details

4.24 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.

4.24.1 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 section 3.2.2 on page 8.

4.24.2 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.

4.25 QTM RT Packet 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:

Size in bytes	Name	Description
4	Size	The total size of the QTM RT packet including these four bytes denoting the size.
4	Type	The type of data in the packet

After the header follows the actual data of the packet:

Size – 8	Data	Whatever data that the Type field says it is.

NB:
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.

4.25.1 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 / Command Response	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.

4.26 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.

Example of an error packet:

Size in bytes	Name	Value
4	Size	31 (8 bytes header + 23 bytes data)
4	Type	0
23	Data	“Command not supported.” The string is laid out like this (always with a NULL char to terminate it): Byte | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12

		Value: C o m m a n d \32 n o t \32 Byte: 13 14 15 16 17 18 19 20 21 22 23 Value: s u p p o r t e d . \0

4.27 Command / Command Response packet

Commands and responses to commands are sent in packets of type 1 (see table above). Command response strings sent from the server are always NULL-terminated but NULL-termination is optional for command strings sent from the clients.

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

Size in bytes	Name	Value
4	Size	20 (8 bytes header + 12 bytes data)
4	Type	1
12	Data	"Version 1.2" The string is laid out like this (with a NULL char to terminate it, which is not required of clients). Byte: 1 2 3 4 5 6 7 8 9 10 11 12 Value: V e r s i o n \32 1 . 2 \0

4.28 XML packet

XML is used to exchange parameters between the server and the client. This has several benefits, including extendibility. Clients should not assume that the XML sent by the server looks exactly like the examples below – there may be any number of other items included as well, and there may be differences in whitespace etc, but if you use a standard XML parser to look for the items you are interested in this will not be a problem. The QTM RT client SDK source code includes source code for a free XML parser (it is actually the same source code used inside QTM to create the XML, so they should work well together).

XML packets follow the same layout as Command/Response packets and Error packets. The packet header is followed by a NULL-terminated ASCII string. Interpret the string as XML according to the following paragraphs. All XML data strings sent from the QTM RT server are enclosed by a block named from the version of the protocol used (`QTM_Parameters_Ver_1.12` 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_1.12>` block. The individual blocks may appear in any order inside this block.

Size in 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, Analog, Force and Image. See 4.28.1 to 4.28.6.

4.28.1 General XML parameters

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

Frequency

The QTM capture frequency.

Capture_Time

The length of the QTM capture, started with the *start* command. The time is expressed in seconds.

Start_On_External_Trigger

Measurement starts on external trigger. The value can be true or false.

Start_On_Trigger_NO

The Start_On_Trigger_NO setting tells QTM if the measurement shall start on external trigger signal from Miqus Sync Unit Trig NO port or the Oqus trigger input. The value can be true or false.

Start_On_Trigger_NC

The Start_On_Trigger_NC setting tells QTM if the measurement shall start on external trigger signal from Miqus Sync Unit Trig NC port. The value can be true or false.

Start_On_Trigger_Software

The Start_On_Trigger_Software setting tells QTM if the measurement shall start on external trigger signal from a software trigger. It can be devices and applications like keyboard, RT clients, telnet command etc. The value can be true or false.

External_Time_Base

Enabled

Enable or disable external time base. Value can be True or False.

Signal_Source

Signal source used for external time base. Possible values:

- Control port
- IR receiver
- SMPTE
- Video sync

Signal_Mode

Possible values:

- Periodic
- Non-periodic

Frequency_Multiplier

Incoming frequency is multiplied by this integer to get the camera frequency. Can be combined with frequency divisor. Value is an integer.

Frequency_Divisor

Incoming frequency is divided by this integer to get the camera frequency. Can be combined with frequency multiplier. Value is an integer.

Frequency_Tolerance

Frequency tolerance in ppm of period time. Value is an integer.

Nominal_Frequency

Nominal frequency used by QTM. If the value is *None*, nominal frequency is disabled. Otherwise the value is a float.

Signal_Edge

Control port TTL signal edge.

- Negative
- Positive

Signal_Shutter_Delay

Delay from signal to shutter opening in micro seconds. Value is an integer.

Non_Periodic_Timeout

Max number of seconds expected between two frames in non-periodic mode. Value is a float.

Processing_Actions

PreProcessing2D

Enable or disable 2D pre-processing action. Value can be True or False.

Tracking

Enable or disable 3D or 2D tracking processing action. Value can be 3D, 2D or False.

TwinSystemMerge

Enable or disable twin system merge processing action. Value can be True or False.

SplineFill

Enable or disable spline fill processing action. Value can be True or False.

AIM

Enable or disable AIM processing action. Value can be True or False.

Track6DOF

Enable or disable 6DOF tracking processing action. Value can be True or False.

ForceData

Enable or disable force data processing action. Value can be True or False.

GazeVector

Enable or disable gaze vector data processing action. Value can be True or False.

ExportTSV

Enable or disable export to TSV processing action. Value can be True or False.

ExportC3D

Enable or disable export to C3D processing action. Value can be True or False.

ExportMatlabFile

Enable or disable export to MATLAB file processing action. Value can be True or False.

ExportAviFile

Enable or disable export to AVI video file processing action. Value can be True or False.

RealTime_Processing_Actions

PreProcessing2D

Enable or disable 2D pre-processing action. Value can be True or False.

Tracking

Enable or disable 3D tracking processing action. Value can be 3D or False.

AIM

Enable or disable AIM processing action. Value can be True or False.

Track6DOF

Enable or disable 6DOF tracking processing action. Value can be True or False.

ForceData

Enable or disable force data processing action. Value can be True or False.

GazeVector

Enable or disable gaze vector data processing action. Value can be True or False.

ExportTSV

Enable or disable export to TSV processing action. Value can be True or False.

ExportC3D

Enable or disable export to C3D processing action. Value can be True or False.

ExportMatlabFile

Enable or disable export to MATLAB file processing action. Value can be True or False.

ExportAviFile

Enable or disable export to AVI video file processing action. Value can be True or False.

Reprocessing_Actions

PreProcessing2D

Enable or disable 2D pre-processing action. Value can be True or False.

Tracking

Enable or disable 3D or 2D tracking processing action. Value can be 3D, 2D or False.

TwinSystemMerge

Enable or disable twin system merge processing action. Value can be True or False.

SplineFill

Enable or disable spline fill processing action. Value can be True or False.

AIM

Enable or disable AIM processing action. Value can be True or False.

Track6DOF

Enable or disable 6DOF tracking processing action. Value can be True or False.

ForceData

Enable or disable force data processing action. Value can be True or False.

GazeVector

Enable or disable gaze vector data processing action. Value can be True or False.

ExportTSV

Enable or disable export to TSV processing action. Value can be True or False.

ExportC3D

Enable or disable export to C3D processing action. Value can be True or False.

ExportMatlabFile

Enable or disable export to MATLAB file processing action. Value can be True or False.

ExportAviFile

Enable or disable export to AVI video file processing action. Value can be True or False.

Camera_System

Type

Type of camera system. Available types are:

- Unknown
- Oqus
- Miqus

Camera

General settings consist of none or several *Camera* blocks, with following content:

ID

Identity of the camera to which the settings apply.

Model

Model of selected camera. Available models are:

- 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 700
- Miqus M1
- Miqus M3
- Miqus M5
- Miqus Sync Unit

Underwater

True if the camera is an underwater camera.

Supports_HW_Sync

True if the camera supports hardware synchronization.

Serial

Serial number of the selected camera.

Mode

Camera mode for selected camera. Available camera modes are:

- Marker
- Marker Intensity
- Video

Video_Mode

Video mode for selected camera. Available video modes are:

- Custom
- 1080p_24hz
- 720p_25hz
- 720p_50hz
- 540p_25hz
- 540p_50hz
- 540p_60hz
- 480_25hz
- 480_50hz
- 480_60hz
- 480_120hz

Video_Frequency

Video capture frequency for selected camera.

Video_Exposure

There are three video exposure times for the selected camera. Current value, min and max value, which sets the boundaries for the exposure time. The values are in micro seconds.

Video_Flash_Time

There are three video flash times for the selected camera. Current value, min and max value, which sets the boundaries for the flash time. The values are in micro seconds.

Marker_Exposure

There are three marker exposure times for the selected camera. Current value, min and max value, which sets the boundaries for the exposure time. The values are in micro seconds.

Marker_Threshold

There are three marker threshold values for the selected camera. Current value, min and max value, which sets the boundaries for the threshold. The values have no unit.

Position

Position and rotation for selected camera. The position is expressed in X, Y Z coordinates. The rotation is presented by a 3x3 rotational matrix, *Rot_1_1 ... Rot_3_3.*

Orientation

QTM 2D camera view orientation for selected camera. Possible values are: 0, 90, 180 or 270 degrees.

Marker_Res

Marker resolution for selected camera. Width and height values are in sub pixels.

Video_Res

Video resolution for selected camera. Width and height values are in sub pixels.

Marker_FOV

Marker field of view for selected camera. Left, top, right and bottom coordinates are in pixels.

Video_FOV

Video field of view for selected camera. Left, top, right and bottom coordinates are in pixels.

Sync_Out/Sync_Out2

Camera settings consist of none, one *Sync_Out* or one *Sync_Out2*

block, with following content:

Mode

Synchronization mode for the selected camera. Available modes:

- Shutter out
- Multiplier
- Divisor
- Camera independent
- Measurement time
- Continuous 100Hz

Value

Integer value with different content depending on the *Mode* setting.

- Shutter out. - Not used
- Multiplier - Multiplier applied to the camera frequency
- Divisor - Divisor applied to the camera frequency
- Camera independent - Camera independent frequency
- Measurement time - Not used
- Continuous 100Hz - Not used

Duty_Cycle

Output duty cycle in percent (float). Only used in multiplier, divisor and camera independent mode.

Signal_Polarity

TTL signal polarity. Not used in Continuous 100Hz mode. Possible values:

- Positive
- Negative

Sync_Out_MT

Camera settings consist of none or one *Sync_Out_MT* block, with following content:

Signal_Polarity

TTL signal polarity. Not used in Continuous 100Hz mode. Possible values:

- Positive
- Negative

Example

```
<QTM_Parameters_Ver_1.15>
    <General>
        <Frequency>240</Frequency>
        <Capture_Time>2.5</Capture_Time>
        <Start_On_External_Trigger>True</Start_On_External_Trigger>
        <Start_On_Trigger_NO>True</Start_On_Trigger_NO>
        <Start_On_Trigger_NC>False</Start_On_Trigger_NC>
        <Start_On_Trigger_Software>False</Start_On_Trigger_Software>
        <External_Time_Base>
            <Enabled>True</Enabled>
            <Signal_Source>Control_Port</Signal_Source>
            <Signal_Mode>Periodic</Signal_Mode>
            <Frequency_Multiplier>1</Frequency_Multiplier>
            <Frequency_Divisor>1</Frequency_Divisor>
            <Frequency_Tolerance>1000</Frequency_Tolerance>
            <Nominal_Frequency>None</Nominal_Frequency>
            <Signal_Edge>Negative</Signal_Edge>
            <Signal_Shutter_Delay>10000</Signal_Shutter_Delay>
            <Non_Periodic_Timeout>10</Non_Periodic_Timeout>
```

```
</External_Time_Base>
<Processing_Actions>
    <Tracking>3D</Tracking>
    <TwinSystemMerge>False</TwinSystemMerge>
    <SplineFill>True</SplineFill>
    <AIM>True</AIM>
    <Track6DOF>False</Track6DOF>
    <ForceData>False</ForceData>
    <ExportTSV>False</ExportTSV>
    <ExportC3D>False</ExportC3D>
    <ExportMatlabFile>False</ExportMatlabFile>
</Processing_Actions>
<Camera_System>
    <Type>Oqus</Type>
</Camera_System>
<Camera>
    <ID>1</ID>
    <Model>Oqus 300</Model>
    <Underwater>False</Underwater>
    <Serial>7658787</Serial>
    <Mode>Marker</Mode>
    <Video_Mode>Custom</Video_Mode>
    <Video_Frequency>30</Video_Frequency>
    <Video_Exposure>
        <Current>10000</Current>
        <Min>5</Min>
        <Max>39980</Max>
    </Video_Exposure>
    <Video_Flash_Time>
        <Current>1000</Current>
        <Min>5</Min>
        <Max>2000</Max>
    </Video_Flash_Time>
    <Marker_Exposure>
        <Current>1000</Current>
        <Min>5</Min>
        <Max>2000</Max>
    </Marker_Exposure>
    <Marker_Threshold>
        <Current>200</Current>
        <Min>50</Min>
        <Max>900</Max>
    </Marker_Threshold>
    <Position>
        <X>100.0>/X>
        <Y>200.0>/Y>
        <Z>100.0>/Z>
        <Rot_1_1>0.0</Rot_1_1>
        <Rot_2_1>0.0</Rot_2_1>
        <Rot_3_1>0.0</Rot_3_1>
        <Rot_1_2>0.0</Rot_1_2>
        <Rot_2_2>0.0</Rot_2_2>
        <Rot_3_2>0.0</Rot_3_2>
        <Rot_1_3>0.0</Rot_1_3>
        <Rot_2_3>0.0</Rot_2_3>
        <Rot_3_3>0.0</Rot_3_3>
    </Position>
```

```
                <Orientation>0</Orientation>
                <Marker_Res>
                    <Width>81920</Width>
                    <Height>65536</Height>
                </Marker_Res>
                <Video_Res>
                    <Width>1280</Width>
                    <Height>1024</Height>
                </Video_Res>
                <Marker_FOV>
                    <Left>0</Left>
                    <Top>0</Top>
                    <Right>1279</Right>
                    <Bottom>1023</Bottom>
                </Marker_FOV>
                <Video_FOV>
                    <Left>0</Left>
                    <Top>0</Top>
                    <Right>1279</Right>
                    <Bottom>1023</Bottom>
                </Video_FOV>
                <Sync_Out>
                    <Mode>Camera independent</Mode>
                    <Value>120</Value>
                    <Duty_Cycle>50.000</Duty_Cycle>
                    <Signal_Polarity>Negative</Signal_Polarity>
                </Sync_Out>
            </Camera>
        </General>
</QTM_Parameters_Ver_1.15>
```

4.28.2 3D XML parameters

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

Note: XML element names can't begin with a number, that's why the element for 3D parameters is called *The_3D*.

AxisUpwards

This parameter 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 parameter 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 labelled trajectories (markers).

Label

Block containing label information.

Name

The name of the label (trajectory).

RGBColor

The color of the label (trajectory), represented by a three byte integer value. Bit 0-7 represents red, bit 8-15 represents green and bit 16-23 represents blue.

Bones

Block containing bone information.

Bone From

The name of the label (trajectory) where the bone starts.

Bone To

The name of the label (trajectory) where the bone ends.

Color

The name of the label (trajectory) where the bone ends.

Example

```
<QTM_Parameters_Ver_1.15>
    <The_3D>
        <AxisUpwards>+Z</AxisUpwards>
        <CalibrationTime>2011.09.23 11:23:11</CalibrationTime>
        <Labels>2</Labels>
        <Label>
             <Name>th12</Name>
             <RGBColor>ff00ff</RGBColor>
        </Label>
        <Label>
             <Name>r_asis</Name>
             <RGBColor>00ffff</RGBColor>
        </Label>
        <Bones>
             <Bone From=”fromName” To=”toName” Color=ffff00/>
        </Bones>
    </The_3D>
</QTM_Parameters_Ver_1.15>
```

4.28.3 6D XML parameters

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

Note: XML element names can't begin with a number, that's why the element for 3D parameters is called *The_6D*.

Bodies

Number of 6DOF bodies.

Body

Block containing 6DOF body information.

Name

The name of the 6DOF body.

RGBColor

The color of the 6DOF body, represented by a three byte integer value. Bit 0-7 represents red, bit 8-15 represents green and bit 16-23 represents blue.

Point

The X, Y and Z coordinate of one of the points that defines the 6DOF body. The body is defined by 3 or more points.

Example

```
<QTM_Parameters_Ver_1.15>
    <The_6D>
        <Bodies>1</Bodies>
        <Body>
            <Name>Body1</Name>
            <RGBColor>ffffff</RGBColor>
            <Point>
                <X>10.5</X>
                <Y>9.5</Y>
                <Z>10.5</Z>
            </Point>
            <Point>
                <X>10.5</X>
                <Y>9.5</Y>
                <Z>10.5</Z>
            </Point>
            <Point>
                <X>10.5</X>
                <Y>9.5</Y>
                <Z>10.5</Z>
            </Point>
        </Body>
```

```
    </The_3D>
</QTM_Parameters_Ver_1.15>
```

4.28.4 Gaze vector XML parameters

In response to the command *GetParameters GazeVector* the QTM RT server will reply with an XML data packet, containing a block called *Gaze_Vector*. See below for the format of this block.

Vector

Block containing gaze vector information.

Name

The name of the gaze vector body.

Frequency

The gaze vector update frequency.

Example

```
<QTM_Parameters_Ver_1.15>
    <Gaze_Vector>
        <Vector>
            <Name>Gaze 1 (L)</Name>
            <Frequency>30</Frequency>
        </Vector>
        <Vector>
            <Name>Gaze 1 (R)</Name>
            <Frequency>30</Frequency>
        </Vector>
        <Vector>
            <Name>Gaze 2</Name>
            <Frequency>60</Frequency>
        </Vector>
    </Gaze_Vector>
</QTM_Parameters_Ver_1.15>
```

4.28.5 Analog XML parameters

In response to the command *GetParameters Analog* the QTM RT server will reply with XML data packet, containing a block called *Analog*. See below for the format of this block.

Device

Block 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.

Unit

Measurement unit.

Range

Min and max analog measurement values.

Label

Channel name. There shall be as many labels as there are channels.

Example

```
<QTM_Parameters_Ver_1.15>
    <Analog>
        <Device>
            <Device_ID>1</Device_ID>
            <Device_Name>Board 1</Device_Name>
            <Channels>4</Channels>
            <Frequency>2400</Frequency>
            <Range>
                <Min>-10.0</Min>
                <Max>10.0</Max>
            </Range>
            <Channel>
                <Label>Channel 1</Label>
                <Unit>volts</Unit>
            </Channel>
            <Channel>
                <Label>Channel 2</Label>
                <Unit>volts</Unit>
            </Channel>
            <Channel>
                <Label>Channel 3</Label>
                <Unit>volts</Unit>
            </Channel>
            <Channel>
                <Label>Channel 4</Label>
                <Unit>microvolts</Unit>
            </Channel>
        </Device>
    </Analog>
</QTM_Parameters_Ver_1.15>
```

4.28.6 Force XML parameters

In response to the command *GetParameters Force* the QTM RT server will reply with XML data packet, containing a block called *Force*. See below for the format of this block.

Unit_Length

Length unit used in the *Force* XML block.

Unit_Force

Force unit used in the *Force* XML block.

Plate

Block containing force plate information.

Plate_ID

Unique ID of the force plate. An integer value starting with 1.

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 frequence 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

Block containing four blocks with the corners of the force plate. Corner1, Corner2, Corner3 and Corner4. Each corner has an X, Y and Z coordinate value.

Origin

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

Channels

Block containing blocks called Channel. One for each analog channel connected to the force plate. Each *Channel* contains *Channel_No* and *ConversionFactor*.

Calibration_Matrix

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

Example

```
<QTM_Parameters_Ver_1.15>
    <Force>
        <Unit_Length>mm</Unit_Length>
        <Unit_Force >N</Unit_Force>
        <Plate>
            <Plate_ID>1</Plate_ID>
            <Analog_Device_ID>1</Analog_Device_ID>
            <Frequency>1000</Frequency>
            <Type>AMTI</Type>
             <Name>First force plate</Name>
            <Length>600.0</Length>
            <Width>400.0</Width>
            <Location>
                <Corner1>
                    <X>1000.00</X>
                    <Y>0.00</Y>
                    <Z>0.00</Z>
                <Corner1>
                <Corner2>
                    <X>1600.00</X>
                    <Y>0.00</Y>
                    <Z>0.00</Z>
                <Corner2>
                <Corner3>
                    <X>1600.00</X>
                    <Y>400.00</Y>
                    <Z>0.00</Z>
                <Corner3>
                <Corner4>
                    <X>1000.00</X>
                    <Y>400.00</Y>
                    <Z>0.00</Z>
                <Corner4>
            </Location>
            <Origin>
                <X>-4.00</X>
                <Y>5.00</Y>
                <Z>-40.00</Z>
            </Origin>
            <Channels>
                <Channel>
                    <Channel_No>1</Channel_No>
                    <ConversionFactor>4623.454</ConversionFactor>
```

```
            </Channel>
            <Channel>
                <Channel_No>2</Channel_No>
                <ConversionFactor>4623.454</ConversionFactor>
            </Channel>
            <Channel>
                <Channel_No>3</Channel_No>
                <ConversionFactor>4623.454</ConversionFactor>
            </Channel>
            <Channel>
                <Channel_No>4</Channel_No>
                <ConversionFactor>4623.454</ConversionFactor>
            </Channel>
            <Channel>
                <Channel_No>5</Channel_No>
                <ConversionFactor>4623.454</ConversionFactor>
            </Channel>
            <Channel>
                <Channel_No>6</Channel_No>
                <ConversionFactor>4623.454</ConversionFactor>
            </Channel>
            <Channel>
                <Channel_No>7</Channel_No>
                <ConversionFactor>4623.454</ConversionFactor>
            </Channel>
            <Channel>
                <Channel_No>8</Channel_No>
                <ConversionFactor>4623.454</ConversionFactor>
            </Channel>
        </Channels>
        <Calibration_Matrix>
            <Rows>
                <Row>
                    <Columns>
                        <Column>1.00</Column>
                        <Column>0.00</Column>
                        <Column>0.00</Column>
                        <Column>0.00</Column>
                        <Column>0.00</Column>
                        <Column>0.00</Column>
                    </Columns>
                </Row>
                <Row>
                    <Columns>
                        <Column>0.00</Column>
                        <Column>1.00</Column>
                        <Column>0.00</Column>
                        <Column>0.00</Column>
                        <Column>0.00</Column>
                        <Column>0.00</Column>
                    </Columns>
                </Row>
                <Row>
                    <Columns>
                        <Column>0.00</Column>
                        <Column>0.00</Column>
                        <Column>1.00</Column>
```

```
                                <Column>0.00</Column>
                                <Column>0.00</Column>
                                <Column>0.00</Column>
                            </Columns>
                        </Row>
                        <Row>
                            <Columns>
                                <Column>0.00</Column>
                                <Column>0.00</Column>
                                <Column>0.00</Column>
                                <Column>1.00</Column>
                                <Column>0.00</Column>
                                <Column>0.00</Column>
                            </Columns>
                        </Row>
                        <Row>
                            <Columns>
                                <Column>0.00</Column>
                                <Column>0.00</Column>
                                <Column>0.00</Column>
                                <Column>0.00</Column>
                                <Column>1.00</Column>
                                <Column>0.00</Column>
                            </Columns>
                        </Row>
                        <Row>
                            <Columns>
                                <Column>0.00</Column>
                                <Column>0.00</Column>
                                <Column>0.00</Column>
                                <Column>0.00</Column>
                                <Column>0.00</Column>
                                <Column>1.00</Column>
                            </Columns>
                        </Row>
                    </Rows>
                </Calibration_Matrix>
            </Plate>
        </Force>
</QTM_Parameters_Ver_1.15>
```

The parameters for force plates follow roughly the standard of the C3D file format (www.c3d.org).

4.28.7 Image XML parameters

In response to the command *GetParameters Image* the QTM RT server will reply with XML data packet, containing a block called *Image*. See below for the format of this block.

Camera

The *Image* block contains one or several *Camera* blocks.

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).

Example

```
<QTM_Parameters_Ver_1.15>
    <Image>
        <Camera>
            <ID>1</ID>
```

```
            <Enabled>False</Enabled>
            <Format>JPG</Format>
            <Width>640</Width>
            <Height>400</Height>
            <Left_Crop>0.0</Left_Crop >
            <Top_Crop>0.0</Top_Crop>
            <Right_Crop>1.0</Right_Crop>
            <Bottom_Crop>1.0</Bottom_Crop>
        </Camera>
    </Image>
</QTM_Parameters_Ver_1.15>
```

4.29 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

Size in bytes	Name	Value/Description
4	Size	8 bytes packet header + 12 bytes data frame header + the size of all the components and their headers. 32-bit integer
4	Type	Value = 3. 32-bit integer
8	Time Stamp	Number of microseconds from start, 64-bit integer. The time stamp value is not valid for the Analog and Force data frame components, they have their own timestamps in their component data.
4	Frame Number	The number of this frame. The frame number is not valid for the Analog and Force data frame components. They have their own frame numbers within the component. 32-bit integer.
4	Component Count	The number of data components in the data packet. 32-bit integer.

Component data (Repeated ComponentCount times)

Size in bytes	Name	Value/Description
4	Component Size	Size of *Component Data* + 8 bytes component header. 32-bit integer.
4	Component Type	The type of the component. Defined in the following section. 32-bit integer.
Size – 8	Component Data	Component-specific data. Defined in the following sections, 4.29.1 to 4.29.2.

4.29.1 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.

Type no	Name	Description
7	2D	2D marker data
8	2D Linearized	Linearized 2D marker data
1	3D	3D marker data
9	3D Residuals	3D marker data with residuals
2	3D No Labels	Unidentified 3D marker data
10	3D No Labels Residuals	Unidentified 3D marker data with residuals
3	Analog	Analog data from available analog devices
13	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.
4	Force	Force data from available force plates.
15	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.
5	6D	6D data – position and rotation matrix
11	6D Residuals	6D data – position and rotation matrix with residuals
6	6D Euler	6D data – position and Euler angles
12	6D Euler Residuals	6D data – position and Euler angles with residuals
14	Image	Image frame from a specific camera. Image size and format is set with the XML settings, see 3.5.2.
16	Gaze Vector	Gaze vector data defined by a unit vector and position.

4.29.2 2D and 2D linearized component

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

Size in bytes	Name	Description
4	Component Size	The size of the component including the header (*Component Size*, *Component Type* and *Camera Count*). 32-bit integer.
4	Component Type	Value 7 or 8. See section 4.29.1. 32-bit integer.
4	Camera Count	Number of cameras. 32-bit integer.
2	2D Drop Rate	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. 16-bit integer.
2	2D Out Of Sync Rate	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. 16-bit integer.

Repeated CameraCount times:

4	Marker Count	The number of markers for this camera in this frame. 32-bit integer
1	Status Flags	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	2D marker data from the camera, described below:

2D Data, repeated MarkerCount times:

4	X	X coordinate of the marker, 32-bit integer.
4	Y	Y coordinate of the marker, 32-bit integer.
2	Diameter X	Marker X size, 16-bit integer.
2	Diameter Y	Marker Y size, 16-bit integer.

4.29.3 3D component

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.

Size in bytes	**Name**	**Description**
4	Component Size	The size of the component including the header (*Component Size*, *Component Type* and *Marker Count*). 32-bit integer
4	Component Type	Value = 1. See section 4.29.1. 32-bit integer
4	Marker Count	The number of markers in this frame. 32-bit integer
2	2D Drop Rate	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. 16-bit integer
2	2D Out Of Sync Rate	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. 16-bit integer.

Repeated MarkerCount times:

4	X	X coordinate of the marker, 32-bit float.
4	Y	Y coordinate of the marker, 32-bit float.
4	Z	Z coordinate of the marker, 32-bit float.

4.29.4 3D with residuals component

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 64 bits set – this signifies a negative quiet Not-A-Number according to the IEEE 754 floating point standard. This frame component is the same as the 3D data frame, except for the residual for each 3D point.

Size in bytes	Name	Description
4	Component Size	The size of the component including the header (*Component Size*, *Component Type* and *Marker Count*). 32-bit integer.
4	Component Type	Value = 9. See section 4.29.1. 32-bit integer.
4	Marker Count	The number of markers in this frame. 32-bit integer.
2	2D Drop Rate	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. 16-bit integer.
2	2D Out Of Sync Rate	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. 16-bit integer.

Repeated *Marker Count* times:

4	X	X coordinate of the marker, 32-bit float.
4	Y	Y coordinate of the marker, 32-bit float.
4	Z	Z coordinate of the marker, 32-bit float.
4	Residual	Residual for the 3D point. 32-bit float.

4.29.5 3D no labels component

Size in bytes	Name	Description
4	Component Size	The size of the component including the header (*Component Size*, *Component Type* and *Marker Count*). 32-bit integer.
4	Component Type	Value = 2. See section 4.29.1. 32-bit integer.
4	Marker Count	The number of markers in this frame. 32-bit integer.
2	2D Drop Rate	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. 16-bit integer.
2	2D Out Of Sync Rate	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. 16-bit integer.

Repeated *Marker Count* times:

4	X	X coordinate of the marker, 32-bit float.
4	Y	Y coordinate of the marker, 32-bit float.
4	Z	Z coordinate of the marker, 32-bit float.
4	ID	An unsigned integer ID that serves to identify markers between frames. 32-bit integer.

4.29.6 3D no labels with residuals component

Size in bytes	Name	Description
4	Component Size	The size of the component including the header (*Component Size*, *Component Type* and *Marker Count*). 32-bit integer.
4	Component Type	Value = 10. See section 4.29.1. 32-bit integer.
4	Marker Count	The number of markers in this frame. 32-bit integer.
2	2D Drop Rate	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. 16-bit integer.
2	2D Out Of Sync Rate	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. 16-bit integer.

Repeated *Marker Count* times:

4	X	X coordinate of the marker, 32-bit float.
4	Y	Y coordinate of the marker, 32-bit float.
4	Z	Z coordinate of the marker, 32-bit float.
4	ID	An unsigned integer ID that serves to identify markers between frames. 32-bit integer.
4	Residual	Residual for the 3D point. 32-bit float.

4.29.7 6DOF component

Size in bytes	Name	Description
4	Component Size	The size of the component including the header (*Component Size*, *Component Type* and *Body Count*). 32-bit integer.
4	Component Type	Value = 5. See section 4.29.1. 32-bit integer.
4	Body Count	The number of 6DOF bodies in this frame. 32-bit integer.

2	2D Drop Rate	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. 16-bit integer.
2	2D Out Of Sync Rate	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. 16-bit integer.

Repeated *Body Count* times:

4	X	X coordinate of the body, 32-bit float.
4	Y	Y coordinate of the body, 32-bit float.
4	Z	Z coordinate of the body, 32-bit float.
9 * 4	Rotation	Rotation matrix of the body, 9 32-bit floats.

4.29.8 6DOF with residuals component

Size in bytes	**Name**	**Description**
4	Component Size	The size of the component including the header (*Component Size*, *Component Type* and *Body Count*). 32-bit integer.
4	Component Type	Value = 11. See section 4.29.1. 32-bit integer.
4	Body Count	The number of 6DOF bodies in this frame. 32-bit integer.
2	2D Drop Rate	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. 16-bit integer.
2	2D Out Of Sync Rate	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. 16-bit integer.

Repeated *Body Count* times:

4	X	X coordinate of the body, 32-bit float.
4	Y	Y coordinate of the body, 32-bit float.
4	Z	Z coordinate of the body, 32-bit float.
9 * 4	Rotation	Rotation matrix of the body, 9 32-bit floats.
4	Residual	Residual for the 6D body. 32-bit float.

4.29.9 6DOF Euler component

Size in bytes	**Name**	**Description**
4	Component Size	The size of the component including the header (*Component Size*, *Component Type* and *Body Count*). 32-bit integer.
4	Component Type	Value = 6. See section 4.29.1. 32-bit integer.
4	Body Count	The number of 6DOF bodies in this frame. 32-bit integer.
2	2D Drop Rate	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. 16-bit integer.
2	2D Out Of Sync Rate	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. 16-bit integer.

Repeated *Body Count* times:

4	X	X coordinate of the body, 32-bit float.
4	Y	Y coordinate of the body, 32-bit float.
4	Z	Z coordinate of the body, 32-bit float.
4	Angle 1	First Euler angle, in degrees, as defined on the Euler tab in QTM's workspace options. 32-bit float.
4	Angle 2	Second Euler angle, 32-bit float.
4	Angle 3	Third Euler angle, 32-bit float.

4.29.10 6DOF Euler with residuals component

Size in	**Name**	**Description**

bytes		
4	Component Size	The size of the component including the header (Component Size, Component Type and *Body Count*). 32-bit integer.
4	Component Type	Value = 12. See section 4.29.1. 32-bit integer.
4	Body Count	The number of 6DOF bodies in this frame. 32-bit integer.
2	2D Drop Rate	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. 16-bit integer.
2	2D Out Of Sync Rate	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. 16-bit integer.

Repeated *Body Count* times:

4	X	X coordinate of the body, 32-bit float.
4	Y	Y coordinate of the body, 32-bit float.
4	Z	Z coordinate of the body, 32-bit float.
4	Angle 1	First Euler angle, in degrees, as defined on the Euler tab in QTM's workspace options. 32-bit float.
4	Angle 2	Second Euler angle, 32-bit float.
4	Angle 3	Third Euler angle, 32-bit float.
4	Residual	Residual for the 6D body. 32-bit float.

4.29.11 Analog component

Size in bytes	**Name**	**Description**
4	Component Size	The size of the component including the header (*Component Size*, *Component Type* and *Analog Device Count*). 32-bit integer.
4	Component Type	Value = 3. See section 4.29.1. 32-bit integer.

4	Analog Device Count	Number of analog devices in this component. 32-bit integer. **NOTE:** If analog device count is 0 and component size is greater than 12 bytes, the analog component contains only channels selected by the user across all analog devices.

Repeated *Analog Device Count* times:

4	Analog Device ID	Id of this analog device. Id starts at 1. 32-bit integer. **NOTE:** If analog device id is 0, the analog component contains only channels selected by the user across all analog devices.
4	Channel Count	The number of channels of this analog device in this frame. 32-bit integer.
4	Sample Count	The number of analog samples per channel in this frame. 32-bit integer.
4	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. 32-bit integer. *Sample Number* is omitted if *Sample Count* is 0.
4 * *Channel Count* * *Sample Count*	Analog Data	Voltage values for all samples of all channels as 32-bit floats. 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 Data* is omitted if *Sample Count* is 0.

4.29.12 Analog single component

Size in bytes	Name	Description
4	Component Size	The size of the component including the header (*Component Size*, *Component Type* and *Analog Device Count*). 32-bit integer.
4	Component Type	Value = 13. See section 4.29.1. 32-bit integer.
4	Analog Device Count	Number of analog devices in this component. 32-bit integer.

Repeated *Analog Device Count* times:

4	Analog Device ID	Id of this analog device. Id starts at 1. 32-bit integer.
4	Channel Count	The number of channels of this analog device in this frame. 32-bit integer.
4 * Channel Count	Analog Data	Voltage values as 32-bit floats, starting with channel 1.

If no analog data is available, *Analog Data* will contain IEEE NaN (Not a number) float values.

4.29.13 Force component

Size in bytes	Name	Description
4	Component Size	The size of the component including the header (*Component Size*, *Component Type* and *Plate Count*). 32-bit integer.
4	Component Type	Value = 4. See section 4.29.1. 32-bit integer.
4	Plate Count	The number of force plates in this frame. 32-bit integer.

Repeated *Plate Count* times:

4	Force Plate ID	Id of the analog device in this frame. Id starts at 1. 32-bit integer.
4	Force Count	Number of forces in this frame. 32-bit integer.
4	Force Number	Order number of first force in this frame. *Force Number* is increased with the force frequency. 32-bit integer.

36 * Force Count	Force Data	Each force sample consists of 9 32-bit float values: 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

4.29.14 Force single component

Size in bytes	Name	Description
4	Component Size	The size of the component including the header (*Component Size*, *Component Type* and *Plate Count*). 32-bit integer.
4	Component Type	Value = 15. See section 4.29.1. 32-bit integer.
4	Plate Count	The number of force plates in this frame. 32-bit integer.

Repeated *Plate Count* times:

4	Force Plate ID	Id of the analog device in this frame. Id starts at 1. 32-bit integer.
36	Force Data	Each force sample consists of 9 32-bit float values: 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, *Force Data* will contain IEEE NaN (Not a number) float values.

4.29.15 Image component

Size in bytes	Name	Description
4	Component Size	The size of the component including the header (*Component Size*, *Component Type* and *Camera Count*). 32-bit integer.
4	Component Type	Value = 14. See section 4.29.1. 32-bit integer.
4	Camera Count	Number of cameras. 32-bit integer.

Repeated *Camera Count* times:

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

4.29.16 Gaze vector

Size in bytes	Name	Description
4	Component Size	The size of the component including the header (*Component Size*, *Component Type* and *Body Count*). 32-bit integer.
4	Component Type	Value = 16. See section 4.29.1. 32-bit integer.
4	Gaze Vector Count	The number of gaze vectors in this frame. 32-bit integer.

Repeated *Gaze Vector Count* times:

4	Sample Count	Number of vector samples in this frame. 32-bit integer.
0 if Sample Count = 0 4 if Sample Count > 0	Sample Number	Order number of first gaze vector sample in this frame. *Sample Number* is increased with the gaze vector frequency. 32-bit integer.
24 * Sample Count	Vector X	X component of the gaze vector, 9 32-bit float.
	Vector Y	Y component of the gaze vector, 9 32-bit float.
	Vector Z	Z component of the gaze vector, 9 32-bit float.
	Position X	X coordinate of the vector, 32-bit float.
	Position Y	Y coordinate of the vector, 32-bit float.
	Position Z	Z coordinate of the vector, 32-bit float.

4.30 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.

Size in bytes	Name	Value
4	Size	8 (only the header is sent) 32-bit integer.
4	Type	Value = 4. 32-bit integer.

4.31 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.

Size in bytes	Name	Value
4	Size	8 header bytes + C3D file size. 32-bit integer.
4	Type	Value = 5. 32-bit integer.
n	C3D file	C3D file

4.32 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.

Size in bytes	Name	Value
4	Size	8 header bytes + C3D file size. 32-bit integer.
4	Type	Value = 8. 32-bit integer.
n	C3D file	C3D file

4.33 Event packet

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

Size in bytes	Name	Value
4	Size	9 bytes. 32-bit integer.
4	Type	Value = 6. 32-bit integer.
1	Event	Event number: 1-13, see 4.33.1.

4.33.1 Events

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

Event number	Name	Comment
1	Connected	QTM connected to camera system or a QTM file was loaded via the rt interface.
2	Connection Closed	QTM disconnected from camera system.
3	Capture Started	QTM started measurement.
4	Capture Stopped	QTM stopped measurement.
5	Not used	
6	Calibration Started	QTM started camera system calibration.
7	Calibration Stopped	QTM stopped camera system calibration.
8	RT From File Started	QTM started sending real-time data from a QTM-file.

9	RT From File Stopped	QTM stopped sending real-time data from a QTM-file.
10	Waiting For Trigger	QTM is waiting for trigger input from user.
11	Camera Settings Changed	User has changed camera settings.
12	QTM Shutting Down	QTM is shutting down.
13	Capture Saved	QTM has saved current measurement.

Connected

This event is sent by the server when QTM has connected to the camera system.

Connection Closed

This event is sent by the server when QTM has disconnected from the camera system.

Capture Started

This event is sent by the server when QTM has started a capture.

Capture Stopped

This event is sent by the server when QTM has stopped a capture.

Fetching Finished

This event is sent by the server when QTM has finished fetching a capture.

Calibration Started

This event is sent by the server when QTM has started a calibration.

Calibration Stopped

This event is sent by the server when QTM has stopped a calibration.

RT From File Started

This event is sent by the server when QTM has started real time transmissions from a file.

RT From File Stopped

This event is sent by the server when QTM has stopped real time transmissions from a file.

Waiting For Trigger

This event is sent by the server when QTM is waiting for the user to press the trigger button.

Camera Settings Changed

This event is sent by the server when the settings have changed for one or several cameras. This event will not show in the GetState command response.

QTM Shutting Down

This event is sent by the server when QTM is shutting down. This event will not show in the GetState command response.

Capture Saved

This event is sent by the server when QTM has saved the current measurement. This event will not show in the GetState command response.

4.34 Discover packet

When this type of packet is broadcasted to QTMs auto discovery port, see 3.2.2, QTM responds with a *discover response packet*, see 4.34.1.

Size in bytes	Name	Value
4	Size	10 bytes. 32-bit integer. Little endian
4	Type	Value = 7. 32-bit integer. Little endian
2	Response Port	Response port number: 0 – 65535, unsigned 16-bit integer. 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.

4.34.1 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.

Size in bytes	Name	Value
4	Size	10 bytes. 32-bit integer. Little endian
4	Type	Value = 1. 32-bit integer. Little endian
n+1	Server info string	Null terminated string containing, server host name, QTM version and number of connected cameras. n = string size.
2	RT server base port.	Base port number: 0 – 65535, 16-bit unsigned integer. 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 3.2.2 to get the desired port number.

5 OPEN SOUND CONTROL

The OSC version of the QTM RT server uses the Open Sound Control 1.0 specification. http://opensoundcontrol.org

5.1 Connecting

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 5.2.1. A connection is closed with the disconnect command, see 5.2.2.

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.

5.1.1 OSC port number

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 3.2.2.

5.2 OSC 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 (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 summary

Connect Port
Disconnect
Version
QTMVersion

GetState	
GetParameters	All | ([General] [3D] [6D] [Analog] [Force] [GazeVector])
GetCurrentFrame	All | ([2D] [2DLin] [3D] [3DRes] [3DNoLabels] [3DNoLabelsRes] [Analog] [AnalogSingle] [Force] [ForceSingle] [6D] [6DRes] [6DEuler] [6DEulerRes] [GazeVector])
StreamFrames	Stop | ((FrequencyDivisor:n | Frequency:n | AllFrames) [UDP[:address]:port] (All | ([2D] [2DLin] [3D] [3DRes] [3DNoLabels] [3DNoLabelsRes] [Analog] [AnalogSingle] [Force] [ForceSingle] [6D] [6DRes] [6DEuler] [6DEulerRes] [GazeVector])))

5.2.1 Connect

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.

5.2.2 Disconnect

Disconnects the client from the QTM RT server.

5.2.3 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 OSC protocol. You can only retrieve current version.

Example: Command: *Version*
Response: *Version is 1.15*

5.2.4 QTMVersion

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

Example: Command: *QTMVersion*
Response: *QTM Version is 2.3 (build 464)*

5.2.5 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: Event data packet with last QTM event.

5.2.6 GetParameters

```
GetParameters All |
  ([General] [3D] [Analog] [Force] [Image] [GazeVector])
```

This command retrieves the settings for the requested component(s) of QTM in XML format. The XML parameters are described in section 4.28 on page 34.

Example: Command: *GetParameters 3D Force*
Response: XML string containing requested parameters

5.2.7 GetCurrentFrame

```
GetCurrentFrame All |
  ([2D] [2DLin] [3D] [3DRes] [3DNoLabels] [3DNoLabelsRes]
  [Analog[:channels]] [AnalogSingle[:channels]]
  [Force] [ForceSingle] [6D][6DRes] [6DEuler] [6DEulerRes]
  [GazeVector])
```

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

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 section 4.29 on page 55.
- 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 (see section 4.30 on page 69).

- 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.

Example: Command: *GetCurrentFrame 3D Analog*
Response: One data frame is sent to the client according to the data frame protocol described in section 4.29 on page 55.

5.2.8 StreamFrames

```
StreamFrame Stop |
  ((FrequencyDivisor:n | Frequency:n | AllFrames) All |
  ([2D] [2DLin] [3D] [3DRes] [3DNoLabels] [3DNoLabelsRes]
  [Analog[:channels]] [AnalogSingle[:channels]]
  [Force] [ForceSingle] [6D][6DRes] [6DEuler] [6DEulerRes]
  [GazeVector]))
```

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

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 section 4.29 on page 55.
- 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 (see section 4.30 on page 69).
- 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 a rate of approximately n Hz.
    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 60Hz. This means that usually every other processed frame is transmitted, but once in a while two frames in a row are transmitted (to reach 60Hz instead of 50).
  - **AllFrames**
    When a client specifies *AllFrames* in the *StreamFrames* command, every real-time frame processed by QTM is transmitted to the client.

- 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 (see section 4.30 on page 69).
- 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.

Example: Command: *StreamFrames Frequency:30 UDP:2234 3D Analog*
Response: 30 frames per second containing 3D data and Analog data are streamed over UDP/IP to the client computer's port 2234. The data frame protocol is described in section 4.29 on page 55.

5.3 OSC Packet structure

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 4.25.

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.

5.3.1 OSC 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.

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.

5.4 OSC Error packet

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.”

5.5 OSC Command packet

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”

5.6 OSC Command response packet

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”

5.7 OSC XML packet

The XML string contains the same data as for the standard XML packet. See 4.28.

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 4.28.

5.8 OSC Data frame 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* 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 5.8.2.

5.8.1 OSC data frame header

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

OSC type	Name	Value
Int32	TimeStamp Hi	Hi 32 bits of 64 bit timestamp value. Number of microseconds from start. The time stamp value is not valid for the Analog and Force data frame components, they have their own timestamps in their component data.
Int32	TimeStamp Lo	Lo 32 bits of 64 bit timestamp value. See above.
Int32	SMPTETimeCode	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	FrameNumber	The number of this frame. The frame number is not valid for the Analog and Force data frame components. They have their own sample numbers in their component data.
Int32	2DDropRate	How many individual camera 2D frames that have been lost, 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.
Int32	2DOutOfSyncRate	How many individual camera 2D frames, in frames per thousand over the last 0.5 to 1.0 seconds, that have not had the same frame number as the other frames. 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.
Int32	ComponentCount	The number of data components in the data message. Each component is sent as a separate OSC message.

5.8.2 OSC Data frame component types

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 5.8.4.
3D Residuals	/qtm/3d_res	3D marker data with residuals. Each marker has its own OSC address. See 5.8.5.5.8.4
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 5.8.12.
6D Residuals	/qtm/6d_res	6D data – position and rotation matrix with residuals. Each body has its own OSC address. See 5.8.13.
6D Euler	/qtm/6d_euler	6D data – position and Euler angles. Each body has its own OSC address. See 5.8.14.
6D Euler Residuals	/qtm/6d_euler_res	6D data – position and Euler angles with residuals. Each body has its own OSC address. See 5.8.15.
Gaze Vector	/qtm/gaze_vector	Gaze vector data – Unit vector and position. Each gaze vector has its own OSC address.

5.8.3 OSC 2D and 2D linearized component

The 2D and 2D linearized data frame format are the same. The only difference is that the coordinates are linearized in 2D linearized. The OSC address of this message is /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.

5.8.4 OSC 3D component

Each marker is sent in a separate OSC message. The OSC address of this message is /qtm/3d/ with the name of the marker in 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.

5.8.5 OSC 3D with residuals component

Each marker is sent in a separate OSC message. The OSC address of this message is /qtm/3d_res/ with the name of the marker in 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.

5.8.6 OSC 3D no labels component

The OSC address of this message is /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.

5.8.7 OSC 3D no labels with residuals component

The OSC address of this message is /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.

5.8.8 OSC Analog component

The OSC address of this message is /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 …

5.8.9 OSC Analog single component

The OSC address of this message is /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.

5.8.10 OSC Force component

The OSC address of this message is /qtm/force.

OSC type	Name	Description
Int32	PlateCount	The number of force plates in this frame.

Repeated PlateCount 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 * 9) float values. 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 Count* = 0 (force not visible in QTM), *Force Number* and *Force Data* is omitted.

5.8.11 OSC Force single component

The OSC address of this message is /qtm/force_single.

OSC type	Name	Description
Int32	PlateCount	The number of force plates in this frame.

Repeated PlateCount times:

OSC type	Name	Description
Int32	Force Plate ID	Id of the analog device in this frame. Starts at 1.
Float32	Force Data	There are (Force Count * 9) float values. 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.

5.8.12 OSC 6DOF component

Each body is sent in a separate OSC message. The OSC address of this message is /qtm/6d/ with the name of the body in 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.

5.8.13 OSC 6DOF with residuals component

Each body is sent in a separate OSC message. The OSC address of this message is /qtm/6d_res/ with the name of the body in 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.

5.8.14 OSC 6DOF Euler component

Each body is sent in a separate OSC message. The OSC address of this message is /qtm/6d_euler/ with the name of the body in 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.

5.8.15 OSC 6DOF Euler with residuals component

Each body is sent in a separate OSC message. The OSC address of this message is /qtm/6d_euler_res/ with the name of the body in 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.

5.8.16 OSC Gaze vector component

Each gaze vector is sent in a separate OSC message. The OSC address of this message is /qtm/gaze_vector/ with the name of the gaze vector in the end of the address string. Example: /qtm/gaze_vector/Eye.

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	X Vector	X component of the gaze unit vector.
Float32	Y Vector	Y component of the gaze unit vector.
Float32	Z Vector	Z component of the gaze unit vector.
Float32	X Position	X coordinate of the gaze vector.
Float32	Y Position	Y coordinate of the gaze vector.
Float32	Z Position	Z coordinate of the gaze vector.

5.9 OSC 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.

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.

5.10 OSC Event Data packet

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

OSC type	Name	Value
OSC-string	Event	Event string. See 5.10.1.

5.10.1 Events

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

Event number	Name
1	Connected

2	Connection Closed
3	Capture Started
4	Capture Stopped
5	Fetching Finished
6	Calibration Started
7	Calibration Stopped
8	RT From File Started
9	RT From File Stopped
10	Waiting For Trigger
11	Camera Settings Changed
12	QTM Shutting Down
13	Capture Saved

Connected

This event is sent by the server when QTM has connected to the camera system or a QTM file was loaded via the rt interface.

Connection Closed

This event is sent by the server when QTM has disconnected from the camera system.

Capture Started

This event is sent by the server when QTM has started a capture.

Capture Stopped

This event is sent by the server when QTM has stopped a capture.

Fetching Finished

This event is sent by the server when QTM has finished fetching a capture.

Calibration Started

This event is sent by the server when QTM has started a calibration.

Calibration Stopped

This event is sent by the server when QTM has stopped a calibration.

RT From File Started

This event is sent by the server when QTM has started real time transmissions from a file.

RT From File Stopped

This event is sent by the server when QTM has stopped real time transmissions from a file.

Waiting For Trigger

This event is sent by the server when QTM is starting to wait for trigger to start a measurement.

Camera Settings Changed

This event is sent by the server when the settings have changed for one or several cameras. This event will not show in the GetState command response.

QTM Shutting Down

This event is sent by the server when QTM is shutting down. This event will not show in the GetState command response.

Capture Saved

This event is sent by the server when QTM has saved current measurement. This event will not show in the GetState command response.

6 TELNET

Telnet is mainly used for testing. The only way to read data using the Telnet protocol is to stream using udp. When streaming via udp, the data will not show in the Telnet session. You have to read the udp data stream in another application.

6.1 Connecting

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 3.2.2.

6.2 Telnet 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 (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 summary

Version
QTMVersion
ByteOrder

GetState	
GetParameters	All | ([General] [3D] [6D] [Analog] [Force] [GazeVector])
StreamFrames	Stop | ((FrequencyDivisor:n | Frequency:n | AllFrames) [UDP[:address]:port] (All | ([2D] [2DLin] [3D] [3DRes] [3DNoLabels] [3DNoLabelsRes] [Analog] [AnalogSingle] [Force] [ForceSingle] [6D] [6DRes] [6DEuler] [6DEulerRes] [Image] [GazeVector])))
TakeControl	[Password]
ReleaseControl	
New	
Close	
Start	[RTFromFile]
Stop	
Load	Filename
Save	Filename [Overwrite]
LoadProject	ProjectPath
Trig	
SetQTMEvent	Label
Quit	

6.2.1 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 OSC protocol. You can only retrieve current version.

Example: Command: *Version*
Response: *Version is 1.15*

6.2.2 QTMVersion

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

Example: Command: *QTMVersion*
Response: *QTM Version is 2.3 (build 464)*

6.2.3 ByteOrder

Returns the current byte order.

Example: Command: *ByteOrder*

Response: *Byte order is little endian* or
Byte order is big endian

6.2.4 GetState

This command makes the RT server send current QTM state.

Example: 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*
Capture Saved

6.2.5 GetParameters

```
GetParameters All | ([General] [3D] [6D] [Analog] [Force]
[Image] [GazeVector])
```

This command retrieves the settings for the requested component(s) of QTM in XML format. The XML parameters are described in section 4.28 on page 34.

Example: Command: *GetParameters 3D Force*
Response: *Parameters not available* or
XML string containing requested parameters

6.2.6 StreamFrames

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

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

This command makes the QTM RT server start streaming data frames in real-time. Points worth noting are:

- The only way to stream frames using the Telnet protocol is via UDP.
- 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 section 4.29 on page 55.
- 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 (see section 4.30 on page 69).
- 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 a rate of approximately n Hz.
          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 60Hz. This means that usually every other processed frame is transmitted, but

once in a while two frames in a row are transmitted (to reach 60Hz instead of 50).

- **AllFrames**
  When a client specifies *AllFrames* in the *StreamFrames* command, every real-time frame processed by QTM is transmitted to the client.

- UDP notes:
  - UDP is the only way to stream data using Telnet.
    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 (see section 4.30 on page 69).
- 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.

Example: Command: *StreamFrames Frequency:30 UDP:2234 3D Analog*
Response: *QTM started streaming data using UDP*

30 frames per second containing 3D data and Analog data are streamed over UDP/IP to the client computer's port 2234. The data frame protocol is described in section 4.29 on page 55.

6.2.7 TakeControl

TakeControl [Password]

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. 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.

Example: Command: *TakeControl x364k6Gt*
Response: *You are now master* or
You are already master or

127.0.0.1 (1832) is already master or
Client control disabled in QTM *or*
Wrong or missing password

6.2.8 ReleaseControl

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

Example: Command: *ReleaseControl*
Response: *You are now a regular client* or
You are already a regular client

6.2.9 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 4.8.

Example: Command: *New*
Response: *Creating new connection* or
Already connected or
The previous measurement has not been saved or closed or
You must be master to issue this command

6.2.10 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 4.8.

Example: Command: *Close*
Response: *Closing connection* or
Closing file or
No connection to close or
You must be master to issue this command

6.2.11 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 4.8.

Example:	Command:	*Start*	
	Response:	*Starting measurement*	or
		Measurement is already running	or
		Not connected. Create connection with new	or
		Starting RT from file	or
		RT from file already running	or
		No file open	or
		You must be master to issue this command	

6.2.12 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 4.8.

Example:	Command:	*Stop*	
	Response:	*Stopping measurement*	or
		No measurement is running	or
		Parse error	or
		You must be master to issue this command	

6.2.13 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 4.8.

Example:	Command:	*Load*	
	Response:	*Measurement loaded*	or
		Missing file name	or

Failed to load measurement		or
Active camera connection exists		or
Current measurement not saved		or
Parse error		or
You must be master to issue this command		

6.2.14 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 (_##).

It is only possible to issue this command if you have the control over the QTM RT interface. See 4.8.

Example:	Command:	*Save*	
	Response:	*Measurement saved*	or
		Measurement saved as 'new filename with counter'	or
		Failed to save measurement	or
		No write access.	or
		Failed to create directory	or
		No measurement to save	or
		You must be master to issue this command	

6.2.15 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. C:\Users\lnn\QTM files\Imagination

It is only possible to issue this command if you have the control over the QTM RT interface. See 4.8.

Example:	Command:	*Load*	
	Response:	*Project loaded*	or
		Missing project name	or
		Failed to load project	or
		Active camera connection exists	or
		Current measurement not saved	or
		Parse error	or
		You must be master to issue this command	

6.2.16 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 4.33.1. It is only possible to issue this command if you have the control over the QTM RT interface. See 4.8.

Example:	Command:	*Trig*	
	Response:	*Trig ok*	or
		QTM not waiting for trig	or
		You must be master to issue this command	

6.2.17 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.

Example:	Command:	*Event test_event*	
	Response:	*Event set*	or
		Event label too long	or
		QTM is not capturing	or
		You must be master to issue this command	

6.2.18 Quit

`Quit`

This command ends the current telnet session.

Example: Command: *Quit*
Response: *Bye bye*

7 HISTORY OF PROTOCOL CHANGES

7.1 Differences between the 1.15 and the 1.14 version

- 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.

7.2 Differences between the 1.14 and the 1.13 version

- Added bone color to 3d XML parameters.
- Added support for new processing action: *2d pre processing*.
- 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.

7.3 Differences between the 1.13 and the 1.12 version

- 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 6 x 6 matrixes.
- Added support for trajectory bones.

7.4 Differences between the 1.12 and the 1.11 version

- 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.

7.5 Differences between the 1.11 and the 1.10 version

- 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).

7.6 Differences between the 1.10 and the 1.9 version

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

7.7 Differences between the 1.9 and the 1.8 version

- Added *Force single* data component.
- Fixed bug in OSC *Analog*, *Analog single* and *Force* data components.
- Fixed bug in OSC *3D no labels* data component.
- Allow capture start via RT server even if camera system isn't calibrated.

7.8 Differences between the 1.8 and the 1.7 version

- 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 2D linearized 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.

7.9 Differences between the 1.7 and the 1.6 version

- 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*.

7.10 Differences between the 1.6 and the 1.5 version

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

7.11 Differences between the 1.5 and the 1.4 version

- 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

7.12 Differences between the 1.4 and the 1.3 version

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

7.13 Differences between the 1.3 and the 1.2 version

- Added *2D Drop Rate* and *2D Out of sync rate* to frame component header for: 3d, 3DRes, 3DnoLabels and 3DnoLabelsRes.

7.14 Differences between the 1.2 and the 1.1 version

- 2Dlin, 3DRes, 3DnoLabelsRes, 6DRes and 6DEulerRes data type components were added.
- CheckLicense command added.
- `<AxisUpwards>` item added to XML parameters for 3D.

7.15 Differences between the 1.1 and the 1.0 version

- 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*.