Version 1.0

EasyTau Remote Interface Version 1.0

Table of Contents

Definitions in this Document

<CR><LF>	ASCII LineFeed/Carriage Return (Char #13#10) End of line
<Space>	ASCII Space (Char #32)
bold	real Input and Output (Comands, Variables, Return Values)
italic	Parameters
[...]	optional Parameters

Remote Interface

General Things

The Remote Protocol is a text based Protocol (like telnet, e.g. one can use any telnet client to test and operate it) using TCP Port 17748.
EasyTau operates as the server.

- On connect the server announce the EasyTau and remote versions, both correlate to output of the **version** command (see there for more informations). The Output on connect are separated by a <Space>. It's the clients opportunity to check if the Remote interfaces version number fits to the requirement of the client software.
  - example:
    EasyTau<Space>**1.4.3500**<Space>**Remote**<Space>**1.0**<CR><LF>**OK**<CR><LF>
  - This document reflect the status of remote interface version 1.0
- The Server allows only one Client to be connected at one time. Further connection attempts are declined with an error message: **Already connected client** x.x.x.x:p (x.x.x.x:p reflects the IP and Port of the connected client.)
- The Client is automatically disconnected if no command arrived at the Server for at least 120 s.
  - To keep the connection alive the Client should send every 60s a keep alive (<CR><LF>) or ask the Server for the Status (**status**).
- All Commands, Variables and Parameter are case-sensitive and UTF-8 encoded.
- Commands from Client to Server ending with <CR><LF> and processed by the server.
- Every Command to the Server produce at least one of the result answers:
  - **OK**<CR><LF>
    Command processed successfully
  - **ERROR**<Space>*number*<CR><LF>
    Error on processing Command, with error number
- Additional Output from a Command appears as lines before the result answer.
  - Multiple Lines answers are separated by <CR><LF>
- Command and it's Parameter are separated by <Space>, Parameter are separated by comma(,) (without additional spaces)
  - **command**<Space>*Parameter1*,*Parameter2*,*Parameter3*<CR><LF>
- Variables are assigned using the equal sign (=)
  - **VARIABLE**=*Value*<CR><LF>
- The Content of Variables can be read using a question mark behind the variable name
  - **VARIABLE?**<CR><LF>
- FluoTime 300 Devices and Fields are handled as Variables, Device and Field are separated by a point:
  - **DEVICE.Field**=Value<CR><LF>
  - **DEVICE.Field?**
- The number of Devices and Fields available depend on the configuration of the FluoTime. The availability of a Field can be tested with a read attempt on the Field or by parsing output of **devicelist** and **fieldlist**.
- Variables can be assigned at any time. Fields can be assigned only if the Remote interface is not in the Busy state. Usually a Field assignment results in a movement of a Fluo Time 300 device which changes the State to Busy as long the device is moving.
- Usually Commands are written in lower case, Variables, Devices and Fields are written in

upper case. (This is not an requirement but usually the case)

- possible Types of Variables and Field:
  - String – text as chars
  - Int – Integer value e.g. 1, 145, 4566, -42
  - Float – Floating Point value with '.' as decimal spacer. scientific notation possible : e.g. -0.09, 461.5e-9, 15E3
  - Bool – bool variable usually True and False but 1 and 0 also possible.

Commands

Keep-Alive

In principle just an empty command to ensure the Client is still functional. The Server has to answer with **OK** after maximal 120 s to ensure the Server is still functional.

Call: <CR><LF>

Parameter: none

Result answer: always answers **OK**

status – Status of EasyTau and FluoTime 300

Call: **status**

Parameter: none

Result answer: always answers **OK**

Values: Status contains two parts (= two lines in Output)

First line reflects the current Status of EasyTau as Text, a client program have to parse these texts to identify the status:

Idle<CR><LF>

Busy<CR><LF>

Error<CR><LF>

The Second part defines how EasyTau reached the status by a number and a human readable text. To detect special relations only the number should be used. The description can be changed without further notice. Number and description are separated by a <Space>.

Description of possible States.

Idle	***EasyTau waits for commands, e.g.: Skript start***
0 Idle	Initial State after bootup or **reseterror**
1 Script successfully finished	Script finished without an error
2 Script stopped	Script is stopped by user (using **break** via Remote interface or Stop Button on EasyTau GUI)
Busy	***EasyTau is working, e.g. Script is running, Device moving***
1 Script compile	Script is compiling
2 Script running	Script successfully compiled, started and still running
3 Measurement running	Measurement running (TCSPC measuring)
4 Wizard running	Wizard is running in EasyTau (Wizard started in EasyTau GUI)
5 Device moving	FluoTime 300 device is moving (either invoked by Remote Interface or by GUI)

Error	***EasyTau or FluoTime 300 found an error. Maybe no further measurements, Device moving or script starts are possible***
-3 Script Compile Error:	Script compilation failed, compiler message follows after the colon.
-4 Script Runtime Error:	Script successfully compiled and started, but in script runtime an error occurred. Runtimer error message follows after the Colon.
-6 Lost Connection	Connection to FluoTime 300 or TCSPC Electronics lost, no further measurements possible.

Example:

Client: **status**<CR><LF>

Server: **Busy**<CR><LF>

Server: **2**<Space>**Script running**<CR><LF>

Server: **OK**<CR><LF>

version – Version informations

Returns the version of EasyTau and Remote interface.

The Remote interface version contains Major and Minor separated by '.'

Major number changes if the remote interface is changed that clients will not work anymore (changed parameter, changed result answers)

If Minor number changes only, compatible changes are applied on the remote interface. Like bugfixes, timing changes, changed Error texts (but not Error numbers) but also additional non vital commands or variables could appear.

This document reflect the remote interface at version '1.0'.

EasyTau version number contains Major, Minor and SVN Number separated by points '.'. This version number is given here for documentation purpose only.

Call: **version**

Parameter: none

Result answer: always answers **OK**

Values:

1. Remoteinterface Version as: **Remote** x.x
2. EasyTau Version as: **EasyTau** x.x.x

Example:

Client: **version**<CR><LF>

Server: **Remote**<Space>**1.0**<CR><LF>

Server: **EasyTau**<Space>**1.4.3500**<CR><LF>

Server: **OK**<CR><LF>

help – List available Commands and Variables

Call: **help**

Parameter: none

Result answer: always answers **OK**

Values: formated list of known commands and variables with short explanation.

Example:

Client: **help**<CR><LF>

Server: **Commands**<CR><LF>

Server: **status Beschreibung**<CR><LF>

Server: … (more Commands)

Server: **Variables**<CR><LF>

Server: **SAMPLE string Beschreibung**<CR><LF>

Server: … (more Variables)

Server: **OK**<CR><LF>

errormessage – Human readable Error text

Returns the human readable error text for an error number

Call: **errormessage** *errornumber*

Parameter: Errornumber – number which got as result answer of a command (Integer)

Result answer: always answers **OK**

Values: Explanation of the error number as text. Unknown Error numbers return: **Unknown Errorcode** *errornumber*

Example:

Client: **errormessage**<Space>**-1**<CR><LF>

Server: **Illegal Command or Variable**<CR><LF>

Server: **OK**<CR><LF>

quit – Disconnect the client

Ends the connection from server to client from server side. This command does not send a result answer, but immediately disconnect the client from the server.

Call: **quit**

Parameter: none

Result answer: none – connection is disconnected.

scriptlist – List available Scripts

Call: **scriptlist**

Parameter: none

Result answer: always answers **OK**

Values: List of registered Scripts in EasyTau available for remote interface.

Example:

Client: **scriptlist**<CR><LF>

Server: **scriptname**<CR><LF>

Server: … (more Scripts)

Server: **OK**<CR><LF>

devicelist – List available FluoTime 300 Devices

Call: **devicelist**

Parameter: none

Result answer: always answers **OK**

Values: Lists FluoTime 300 devices

Example:

Client: **devicelist**<CR><LF>

Server: **SAMPLE_CHANGER**<CR><LF>

Server: … (more devices)

Server: **OK**<CR><LF>

fieldlist – List available Fields of a Device

Call: **fieldlist** *Devicename*

Parameter: Devicename – a FluoTime 300 device from the Device list (see **devicelist**)

Result answer: possible answers

OK – legal Device name, Fields are printed before OK

ERROR – An Error occured, possible error codes:

-8	Illegal parameter (unknown Device name)

Values: List of available field for the device.

Example:

Client: **fieldlist SAMPLE_CHANGER**<CR><LF>

Server: **SMPPos**<CR><LF>

Server: … (more Fields)

Server: **OK**<CR><LF>

fieldlinfo – Detailed Field Information

Prints formated additional information about the device and it's Field.

Call: **fieldinfo** *Devicename.Fieldname*

Parameter: Devicename.Fieldname – device and field separated by a point '.' (see **devicelist**, **fieldlist**)

Result answer: possible answers:

OK – legal Device and field name, Information are printed before OK

ERROR – An Error occured

-8	Illegal parameter (unknown Device/Field combination)

Values:

Device: Device name

Desc: Device description as shown in EasyTau customized mode

Field: Field name

Desc: Field description as shown in EasyTau customized mode

Type: Field type (Int, Float, Bool, String)

Range: Minimum and Maximum, if the parameter has a range (Int, Float)

Values: Describtion of True/False for Bool Values (Bool) e.g. Open/Closed

Example:

Client: **fieldinfo SAMPLE_CHANGER.SMPPos**<CR><LF>

Server: **Device: SAMPLE_CHANGER**<CR><LF>

Server: **Desc: Sample changer**<CR><LF>

Server: **Field: SMPPos**<CR><LF>

Server: **Desc: Sample pos.**<CR><LF>

Server: **Type: Int**<CR><LF>

Server: **Range: 1 - 4**<CR><LF>

Server: **OK**<CR><LF>

exec – Start a registered Script

Starts a Script. The Command returns immediately if the script was found. Use **status** afterwards to check if it is compiled, started and ended correctly or an error occurred in the runtime. Please note, in contrast to EasyTau for starting a script a Workspace mut be opened before.

Call: **exec** *scriptname*

Parameter: scriptname – a Skript from the Script list (siehe **scriptlist**)

Result answer: possible answers:

OK – Script found and started to compile

ERROR – An Error occurred, possible error codes:

-2	Script not found
-5	Script/Measurement running
-6	FT300/TCSPC is offline
-7	No Workspace open

Example:

Client: **exec** *scriptname*<CR><LF>

Server: **OK**<CR><LF>

break – Stop a running Script

Script is stopped at next possible position. The command returns **OK** immediately, that means the script is still running after this call. Use **status** to check when the script is successful stopped before starting an other script. (**status** → 3 Script stopped) If no script is running, this command has no effect.

Call: **break**

Parameter: none

Result answer: always answers **OK**

Example:

Client: **break**<CR><LF>

Server: **OK**<CR><LF>

reseterror – Removes last error message

If a script runs into an error (compilation or runtime) the EasyTau remote interface changes to error status. This error can be used to change the status back to Idle status. Calling this command in any other status has no effect. Attention: some error states are not possible to reset or will appear directly again. (like Lost Connection)

Call: **reseterror**

Parameter: none

Result answer: always answers **OK**

Example:

Client: **reseterror**<CR><LF>

Server: **OK**<CR><LF>

newworkspace – Create a new Workspace

Creates a new Workspace at the given Path. If the Folder does not exists, the Folder gets created. In this Folder a Log file is created with the same name as the Folder and extension '.etw'. The New Workspace is open automatically. Use **WORKSPACEPATH?** to determine the path of the currently opened Workspace. The given path have to be on the Server computer (EasyTau) or in the network reachable for EasyTau Remote Server.

Call: **newworkspace** *path*

Parameter: path – a path and name for a new Workspace

Result answer: possible answers:

OK – Workspace created and open

ERROR – Error occurred, possible error codes:

-7	No Workspace open, cannot create Workspace

Example:

Client: **newworkspace** *t:\pub\NewWS*<CR><LF>

Server: **OK**<CR><LF>

Client: **WORKSPACEPATH?**<CR><LF>

Server: **t:\pub\NewWS\NewWS.etw**<CR><LF>

Server: **OK**<CR><LF>

openworkspace – Open a Workspace

Opens a previously created Workspaces. Use **WORKSPACEPATH?** to determine the path of the currently opened Workspace. The given path have to be on the Server computer (EasyTau) or in the network reachable for EasyTau Remote Server.

Call: **openworkspace** *path*

Parameter: path – a Path to an existing Workspace, the directory or the log file (.etw) inside

Result answer: possible answers:

OK – Workspace found and open

ERROR – Error occurred, possible error codes:

-7	No Workspace open, cannot open Workspace

Example:

Client: **openworkspace** *t:\pub\NewWS*<CR><LF>

Server: **OK**<CR><LF>

Client: **WORKSPACEPATH?**<CR><LF>

Server: **t:\pub\NewWS\NewWS.etw**<CR><LF>

Server: **OK**<CR><LF>

Variables

SAMPLE – Sample name

Spaces within the sample name are allowed, the complete string until <CR><LF> is used as Sample name. The Sample name is used for the next script start. If no sample name is supplied, “undefined” is used. The Variable can be change during a script is running but the new sample name is used for next script start.

ATTENTION: The sample name can be overwritten within the script

Type: String

Example:

Client: **SAMPLE**=*Rhodamin 6G*<CR><LF>

Server: **OK**<CR><LF>

Client: **SAMPLE**?<CR><LF>

Server: Rhodamin 6G<CR><LF>

Server: **OK**<CR><LF>

SOLVENT – solvent name

Spaces within the solvent name are allowed, the complete string until <CR><LF> is used as solvent name. The solvent name is used for the next script start. If no sample name is supplied, “undefined” is used. The Variable can be change during a script is running but the new solvent name is used for next script start.

ATTENTION: The solvent name can be overwritten within the script.

Type: String

Example:

Client: **SOLVENT**=*Ethanol*<CR><LF>

Server: **OK**<CR><LF>

Client: **SOLVENT**?<CR><LF>

Server: Ethanol<CR><LF>

Server: **OK**<CR><LF>

CONTAINER – Container name (Filename)

The container name should only contain characters allowed as filename. The script can use this name as the Container name. The Variable can be change when a script is running but the new container name is used for next script start. If there exists already a container with this name a

number “(x)” will be added at the end of the name.

Type: String

Example:

Client: **CONTAINER**=*Meas_01_20160101_1455*<CR><LF>

Server: **OK**<CR><LF>

Client: **SOLVENT**?<CR><LF>

Server: Meas_01_20160101_1455<CR><LF>

Server: **OK**<CR><LF>

COMMENT – Comment for next measurement

Spaces within the comment are allowed, the complete string until <CR><LF> is used as solvent name. If the comment should be multi line add a **\n** as new line marker. The comment is used for the next script start. During a script is running the Variable can be changed but the new comment is used for the next script start.

ACHTUNG: The comment can be overwritten within the script

Type: String

Example:

Client: **COMMENT**=A test comment*n*with two lines<CR><LF>

Server: **OK**<CR><LF>

Client: **COMMENT**?<CR><LF>

Server: A test comment<CR><LF>

Server: with two lines<CR><LF>

Server: **OK**<CR><LF>

WORKSPACEPATH – currently open Workspace Path

This value in readonly. An assignment to this value will be ignored.

Type: String

Example:

Client: **WORKSPACEPATH**?<CR><LF>

Server: C:\users\My Documents\ETWorkspace<CR><LF>

Server: **OK**<CR><LF>

Error codes

-1	Illegal command oder variable
-2	Script not found
-3	Script has compilation errors
-4	Script runtime error
-5	Script/Measurement running
-6	FT300/TCSPC is offline
-7	No Workspace open
-8	Wrong parameter
-9	FluoTime 300 busy

Status changes plot