Remote Control Application Programming Interface (API) Communication Protocol

Protocol Version 1.19 (Software Version 2.6)

Revision History

Author	Date	Version	Description
Martin Zaleta	2014-01-17	1.0	Initial draft
Martin Zaleta	2014-01-29	1.1	API tested, optimizations committed
Martin Zaleta	2014-06-19	1.2	Service definitely renamed to API (formerly just Remote-Control Server, or RC Server)
Martin Zaleta	2014-09-09	1.3	Simplified marker detection command, fixed description of %PROBE% parameter
Petr Stolar	2014-09-25	1.4	How to activate the API protocol
Martin Zaleta	2015-06-24	1.5	Revision, update for application version 2.4, more safety precautions
Martin Zaleta	2015-06-29	1.6	Precautions regarding calibration projects
Martin Zaleta	2015-09-23	1.7	Implemented image and video saving, precautions regarding recorded data playback
Martin Zaleta	2015-10-01	1.8	Implemented data export and precautions regarding projects without any scenes
Martin Zaleta	2015-10-05	1.9	Implemented recomputation and notification of stop
Martin Zaleta	2015-10-06	1.10	Minor tweaks of used terms
Martin Zaleta	2015-11-13	1.11	Enhanced LOAD command for easier switching of measurements
Martin Zaleta	2015-11-30	1.12	Further clarifications regarding some commands
Martin Zaleta	2015-12-14	1.13	Licensing limitations considered
Martin Zaleta	2016-05-06	1.14	Slightly tweaked LOAD command
Martin Zaleta	2016-09-02	1.15	Tweaked GETIMAGE and GETVIDEO commands
Martin Zaleta	2016-09-05	1.16	Added LISTPROJECTS command
Martin Zaleta	2017-05-29	1.17	Added a mention about a possible configuration of output values
Martin Zaleta	2017-09-12	1.18	Added an example sequence of commands for a better comprehension
Martin Zaleta	2017-10-18	1.19	Added RESETPROBES command

- The API protocol allows the application to be controlled remotely, either via TCP or COM port serial communication.
- The service implementing the API protocol is activated by adding a new I/O service in the “I/O Services” tab of the application’s main window, selecting the “Remote Control API (TCP/IP, COM port)” option. After the desired connection type is specified, the communication settings and the settings of the output format can be found on the last page of the wizard.
- A successful activation of the API service is indicated by a green circle in the “Active Services” list. A red circle indicates an error which can be displayed by hovering the mouse over the name of the service in the list.

- The communication is performed by sending commands of the format %CMD% [%ARG%] through the specified communication channel, where %CMD% denotes the identifier of the requested command and %ARG% represents the argument(s) required by certain commands.
- The path specified as an argument of certain commands uses a Windows-specific syntax, so an argument of "C:\Program Files\project.mpr" represents a valid path.
- The format for characters used both in sent and received messages is ASCII.
- The individual messages have to be separated by CR or LF or a sequence of both (0x0D 0x0A in ASCII, "\r\n" when using escape characters). It is also recommended to use such separator as a prefix of the first sent command in order to filter out any undesired characters received by the communication channel prior to using this service (ex. from Telnet).
- Whitespace characters and arguments which are not specified by this protocol are ignored.
- The commands are case sensitive and specified as following:

Command identifier	**Description**

LISTPROJECTS [%PATH%]	• Retrieves a list of project files located on the host computer under the path specified as an argument of this command.
	• If no path is specified, it is considered as the one, where the current project is located. This lists all project associated with the current one, itself included.
	• If the specified path is not rooted (absolute), it is considered as relative to the directory of the current project.
	• The names retrieved this way are separated by a CR LF sequence and contain the project file extension in order to avoid possible confusion with the state notifications of the API service.
	• All retrieved names are fully compatible with the LOAD command, as it is valid to specify any of them as its argument.
	• The command is only valid if:
	  ○ A project is open, **unless** the specified path is rooted (absolute).
	  ○ The application is in a responsive state.

LOAD %PATH%	• Opens a project file located on the host computer and sets the loaded project as current.
	• Any changes in the previously open project are saved.
	• The path to the project file is specified as an argument of this command, which is mandatory.
	• If the specified path is not rooted (absolute), it is considered as relative to the directory of the current project.
	• Additionally, if the specified path does not end with an extension specific to a project file, it is added automatically.
	• *Example:* A command of LOAD d08_A5 switches to a project d08_A5.mpr, which is located in the same directory as the current project.
	• This command is only valid if:
	  ○ A project is open, **unless** the specified path is rooted (absolute).
	  ○ A measurement is **not** running.
	  ○ The application is in a responsive state.

START	• Starts a live measurement in the currently open project, which

%MODE%	follows the setup applied to that project. This includes the configuration of a live measurement in the main window, the placement of probes in camera windows, computation settings, camera synchronization parameters and others.
	• Any previously saved frames in the project data directory are overwritten without a prompt.
	• The argument of this command is mandatory and represents the way in which the measured values are sent to the output of the API service. Other active I/O services are unaffected by this choice. It can be one of the following:
	  o AUTO – the values are sent immediately when they are measured and/or computed.
	  o MANUAL – only actual values are sent on-demand using the "GETVALS" command. This is equivalent to starting the live measurement manually using the "Run" button in the main window of the application.
	• This command is only valid if:
	  o A project is open.
	  o A measurement is **not** running.
	  o The recorded data playback is **not** running.
	  o The other usual conditions for the execution of a live measurement are met. (see GUI of the application)
	  o The corresponding function in the GUI of the application is **not** disabled due to licensing limitations.
	  o The application is in a responsive state.

RECOMPUTE	• Starts an offline measurement in the currently open project with recorded frames.
	• During the measurement, only actual values are sent on-demand using the "GETVALS" command. This is equivalent to starting the offline measurement manually using the "Recompute" button in the main window of the application.
	• This command is only valid if:
	  o A project is open.
	  o A measurement is **not** running.
	  o The recorded data playback is **not** running.
	  o The other usual conditions for the execution of an offline measurement are met. (see GUI of the application)
	  o The corresponding function in the GUI of the application is **not** disabled due to licensing limitations.
	  o The application is in a responsive state.

STOP	• Requests to stop the currently running measurement.
	• The measurement is not stopped immediately, so the evaluation of further commands is postponed until an actual stop indicated by a "STOPPED" notification takes place.
	• This command is only valid if:
	  o A project is open.
	  o A measurement is running.
	  o The application is in a responsive state.

GETVALS	• Retrieves the last known measured values. • This command is only valid if: o A project is open.

	○ A measurement is currently running in manual (on-demand) mode or was started manually using the GUI of the application. ○ The application is in a responsive state.
GETIMAGE [%SCENEID%] [%CAMID%] %PATH%	• Saves the currently displayed image in a specified camera display to an image file. • The ID number of the scene, from which the image is to be taken, is specified as the first argument of this command, which is mandatory if there is more than a single scene available in the currently open project. • "Scene" is a common term for either a Mono Camera comprised of single camera display, a Stereo Camera comprised of two camera displays or a Mono Stitching comprised of N camera displays. • The ID number of the camera, from which the image is to be taken, is specified as the second argument of this command, which is mandatory if there is more than a single camera display available in the specified scene. • In a stereo scene, camera IDs of 0 and 1 represent the left and right camera displays respectively, while an ID of 2 represents a special case of a 3D graph display, which can also be accessed by this command. • Likewise, in a stitched scene, there is a special case of an ID, which provides access to the merged camera display, and it is specified as the number of cameras in the scene. (the merged display of a Mono Stitching with three cameras has an ID of 3, whereas IDs of 0, 1 and 2 represent displays of individual cameras) • The path to the destination image file is specified as the last argument of this command, which is mandatory. • *Example:* A command of GETIMAGE 1 2 "C:\Temp\image.png" saves the displayed image from Camera 2 of Scene 1 (which is a Mono Stitching) to a new file at the specified location. • This command is only valid if: ○ A project is open. ○ The current project contains any camera display. ○ The application is in a responsive state.
GETVIDEO [%SCENEID%] [%CAMID%] %PATH%	• Saves an entire recorded range of images in a specified camera display to a video file. • The arguments of this command are specified in the same way as those of GETIMAGE command (see above). • The parameters of the recording process are specified manually in "System Settings" under "Recorded Data Playback". • The range of recorded frames is specified by the slider at the bottom of the main window. • The format of the saved video file is always AVI. • This command is only valid if: ○ A project is open. ○ The current project contains recorded data. ○ The current project contains any camera display. ○ A measurement is **not** running. ○ The recorded data playback is **not** running. ○ The application is in a responsive state.

EXPORT %PATH%	• Exports all computed data in the selected range of the main seeker to a specified csv file.
	• The path to the output file is specified as an argument of this command, which is mandatory.
	• The output csv file uses a standard format with ';' as the field separator.
	• This command is only valid if:
	  ○ A project is open.
	  ○ The current project contains recorded data.
	  ○ A measurement is **not** running.
	  ○ The application is in a responsive state.

DETECT	• Detects markers on the displayed frame in all open scene windows.
	• The parameters of detection are specified manually in "Project Settings" under "Marker Detection" or in a specialized "Detection Settings" dialog.
	• The detected markers are added to the camera displays as point probes, keeping all previously input probes there as well.
	• If it is desired to use line probes comprised of pairs of detected markers instead of point probes, use either the "Automatic Pairing" option which is a part of the detection parameters, or pair the point probes manually using the "Pair Probes" operation.
	• If there are already any computed data present in the open project, they are cleared and the reference is unlocked.
	• This command is only valid if:
	  ○ A project is open.
	  ○ The current project is **not** a calibration project.
	  ○ The current project contains any camera display.
	  ○ A measurement is **not** running.
	  ○ The corresponding function in the GUI of the application is **not** disabled due to licensing limitations.
	  ○ The application is in a responsive state.

CLEAR	• Clears all probes from open camera displays.
	• This operation is **not** revertible.
	• If there are already any computed data present in the open project, they are cleared as well and the reference is unlocked.
	• This command is only valid if:
	  ○ A project is open.
	  ○ The open project is **not** a calibration project.
	  ○ The current project contains any camera display.
	  ○ A measurement is **not** running.
	  ○ The corresponding function in the GUI of the application is **not** disabled due to licensing limitations.
	  ○ The application is in a responsive state.

RESETPROBES	• Resets the position of all probes, which are positioned according to the displayed image. • This currently applies only to neck gauges, whose cross sections are snapped to the edges of the specimen this way. This is the same as if moving the neck gauge manually in the camera display. • If there are already any computed data present in the open project, they are cleared and the reference is unlocked. • This command is only valid if:

	○ A project is open. ○ The current project contains any camera display. ○ A measurement is **not** running. ○ The application is in a responsive state.

- Following the processing of these commands, the application sends a status notification through the specified communication channel.
- The application currently supports only one active TCP communication, so only the client which sent the last request is considered as active.
- The status notifications are separated by a CR LF escape sequence (same as the incoming messages) and can be one of the following:

Notification identifier	Description
OK	The processing of the received command has been successfully completed.
STOPPED	The measurement controlled by "START", "RECOMPUTE" and "STOP" commands has stopped.
ERROR	An error was encountered during the processing of the received command.
INVALID	The received command is invalid in the context of the current application state (see the table of commands for explanations on when a command is valid).
UNKNOWN	The received command is not specified by this protocol.

- Similarly, the application transmits messages containing the measured values through the specified communication channel, also separated by a CR LF escape sequence.
- The format of the messages containing processed data is specified as %VAL1%[|%VAL2%|%VAL3%|…|%VALx%], where %VALx% represents a single measured value corresponding to the x-th data unit on the scrollable results display. Please note, that this is where all the measured values are displayed at once, including the values from all scenes, expressions and even the connected test rig if such is available.
- Example: If three graph data units are measured in a single scene along with two graph data units in another scene, all of their values related to a specific moment in time would be contained in a single message such as this:
  0.164137684065307|0.146714840244903|0.114993578954794|0.0119142302938354|0.01
  01751090343257\r\n
- Alternatively, it is possible to configure the API service to send only explicity specified values to the output. This is done via the "Output Values Settings" dialog, which is accessible through a button labeled "I/O" located next to each value on the scrollable results display.
- The format of the output messages is adjustable in the wizard accessible through the "I/O Services" tab of the main window. It is therefore possible for different instances of the application to use a different format of these messages. The format of the message described in the previous example is the default one.
- The following is an example valid sequence of commands exchanged between the client and server (MercuryRT®) implementing the API protocol. It demonstrates a fully automatized work with the program without any need to interfere with its actual UI. Also included are the

mandatory message separators (\r\n) and comments enclosed in round brackets, which are not a part of the communication:

Client messages	**Server responses**
LISTPROJECTS D:\Data\r\n *(no project is open at the time, so an absolute path is needed)*	project1.mpr\r\n project2.mpr\r\n project3.mpr\r\n OK\r\n
LOAD D:\Data\project1.mpr\r\n *(still no project open, so a project is loaded from an absolute path)*	OK\r\n
CLEAR\r\n *(clears markers left over from a previous work with the project)*	OK\r\n
DETECT\r\n *(performed when the specimen is in frame, detects markers on it)*	OK\r\n
START MANUAL\r\n *(starts a new measurement using the detected markers)*	OK\r\n
GETVALS\r\n *(check the measured values, repeat as many times as necessary)*	3.14|2.71\r\n OK\r\n
GETIMAGE D:\Data\project1.png\r\n *(captures an image right during the measurement for reference)*	OK\r\n
STOP\r\n *(stops the measurement when it is decided for it to be over)*	OK\r\n *(time elapses)* STOPPED\r\n
RECOMPUTE\r\n *(fills in the gaps left during the live measurement, stops by itself)*	OK\r\n *(time elapses)* STOPPED\r\n
GETVIDEO D:\Data\project1.avi\r\n *(captures a video for a detailed report)*	OK\r\n
EXPORT D:\Data\project1.csv\r\n *(exports all the measured values in addition to the video)*	OK\r\n
LOAD project2.mpr\r\n *(an absolute path is not needed anymore and the same process can be repeated over again for all the other prepared projects …)*	…
