STUNNER

**Stunner**
**API Guide**

Issued By:

UNCHAINED LABS

Unchained Labs
6870 Koll Center Parkway Pleasanton, CA 94566

T: +1 (925) 587 9800



**Limited Warranty**

**Instrument Warranty:** Unchained Labs warrants that the instrument purchased will be free from defects in materials and workmanship during the applicable warranty period, provided that the instrument has been operated in accordance with instructions under the supervision of or by a certified operator who has been trained by us. The warranty period for the Instrument terminates twelve (12) months after date of shipment. Within this period, Unchained Labs will, at its sole option, repair or replace any components that fail in normal use. Such repairs or replacement will be made at no charge to the customer for parts and labor. Unchained Labs will cover depot service repairs, if appropriate for the instrument, with parts, labor, and return shipment. This warranty does not cover failures if Unchained Labs has determined that the buyer has misused, improperly installed, maintained, altered, or made unauthorized repairs to the instrument. In addition, this warranty does not apply to wear items, including lamps, unless failure has occurred prior to expected lifetime due to a defect in materials or workmanship.

EXCEPT FOR THE WARRANTIES AND REMEDIES CONTAINED HEREIN, NO OTHER WARRANTIES OR REPRESENTATIONS ARE EXPRESS OR IMPLIED, INCLUDING ANY LIABILITY ARISING UNDER ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, STATUTORY OR OTHERWISE. IN NO EVENT SHALL UNCHAINED LABS BE LIABLE FOR ANY INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, WHETHER RESULTING FROM THE USE, MISUSE, OR INABILITY TO USE THIS PRODUCT OR FROM DEFECTS IN THE PRODUCT.

# TABLE OF CONTENTS

# 1 Introduction

The Stunner is designed for measuring the UV/Vis absorption spectrum of microliter-sized samples, supplemented with photon correlation spectroscopy, designed to read the Stunner Plate.

There is a "**stand-alone**" use of the instrument. The Stunner is then operated by the Lunatic & Stunner Client software, running on a host PC. This host PC communicates with the instrument using a USB cable.

There is also an "**integrated**" way of using the instrument. This requires the instrument to be connected to the network by an Ethernet cable.

Note that the user aspects of the Lunatic & Stunner Client software are described in detail in the Stunner Software User Guide; this includes setting up experiments and retrieving the measurement results. This guide purely concentrates on the integration aspects.

The Unchained Labs Stunner Plates are microfluidic plastic consumables that allow fast and reliable quantification of 2 µl liquid samples, supplemented by particle sizing and aggregation detection. Each plate contains 6 identical transparent pre-mounted Stunner Plate Strips and is designed for high-throughput use. This consumable is ideal for automated workflows as it is easily stackable and includes a barcode for sample tracking. The Stunner Plate is compatible with liquid handling robots (SBS standards) and it allows easy sample loading, preservation and measurement.

This manual describes the integration of the Stunner (including the Lunatic & Stunner Client software) in automated lab setups, for example integration with liquid handling robots. This includes:

- **Physical integration**: description of the mechanical references for integration of the instrument on the work bench.
- **Software integration**: the application programming interface (API) for controlling the Lunatic & Stunner Client software from another program or scripting environment.

# 2 Physical integration

The Stunner is designed for integration with liquid handling robots:

- The consumable is designed for automatic dispensing of the sample by liquid handling robots:
    - Dimensions are compatible with SBS standard (input wells at 9 mm pitch)
    - Self-filling of the sample in the microfluidic meander reduces evaporation, allowing longer storage times of the filled Stunner Plate without influencing the concentration measurements.
- The mechanical design of the consumable and the in/out tray of the instrument are compatible with the grippers of liquid handling robots
- The instrument has mechanical features for indexing the position of the instrument on the work bench.

## 2.1 Integration requirements

For initial configuration

- A PC running Lunatic & Stunner Client, connected to the Stunner via a USB cable.

During general usage:

- A Stunner, connected via Ethernet cable to a network.
- A PC running the API, connected to the same network.

## 2.2 Compatibility with liquid handling robots

Figure 1: Liquid handling robots moving the Stunner Plate in the workbench.

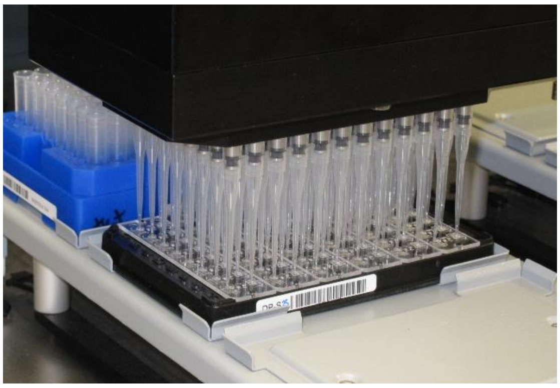


Figure 2: A liquid handling robot filling the 96 input wells of the Stunner Plate.

The Stunner Plate is loaded into the instrument via the in/out tray (Figure 3). Plates on the in/out tray can easily be handled by common robotic arms. Note the alignment mark at the A1 corner of the Stunner Plate: this defines the orientation of the Stunner Plate in the instrument (Figure 4).

Figure 3: Stunner Plate on the in/out tray of the Stunner.

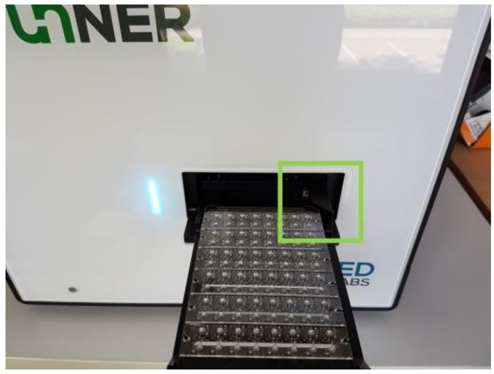


Figure 4: A1 corner alignment mark

The outer dimensions of a Stunner Plate are compatible with standard 96-well plates: width and length of the plate is 85.48 mm x 127.76 mm. The height of the Stunner Plate Frame is 10 mm, and on top of the frame the 1.54 mm thick microfluidic consumable (i.e., the Stunner Plate Strip) is mounted.

Table 1 summarizes the weight of the consumables, as function of the configuration:

| Configuration | Weight |
|---|---|
| 6x Stunner Plate Strips on molded frame | 38g |

Table 1: Weight of consumables.

The input wells are at the same positions as the input wells of a standard 96well plate (as defined in the SBS standard). The shape of the input well is shown in detail in Figure 5.

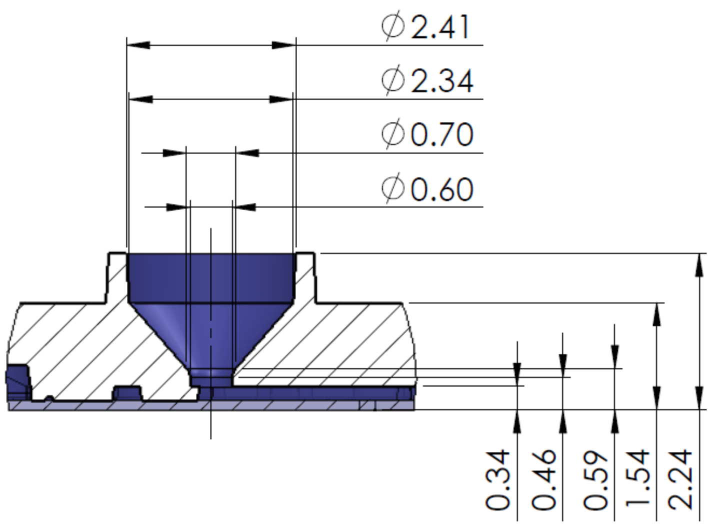



Figure 5: Dimensions (mm) of the input well.

Requirements for optimal dispensing of the sample in the input well:

- Avoid contact of the needle with the input well during dispensing.
- Bring the droplet in contact with the input well during dispensing.

With these guidelines, the 2 µl sample will be transferred directly into the input meander..

## 2.3 Barcode

The instrument includes an internal barcode scanner for reading the barcode. This requires that the barcode label is positioned at a specific location on the skirt of the Stunner Plate: the barcode is attached on the long side of the plate, side of row H, position from column 3 to (including) column 6. See drawing in Figure 6. The scanner reads standard codes, type is CODE128 and, pitch is 2 pixels per line, 14 characters total.

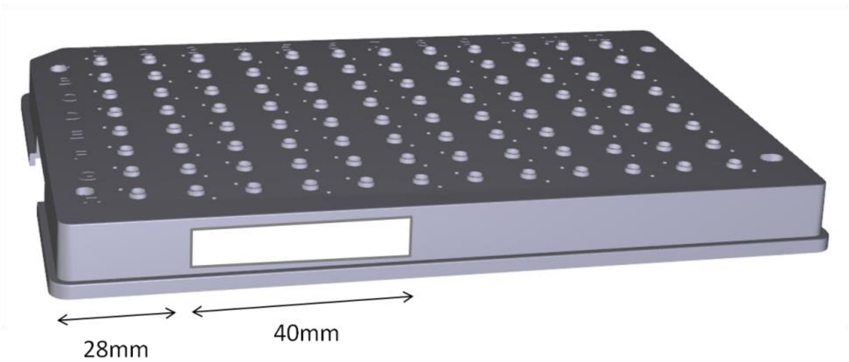



Figure 6: Location of the barcode that can be read by the internal barcode scanner in the instrument.

## 2.4 Mechanical references on the instrument

The dimensions of the instrument are summarized in Appendix 3.

In most lab setups, the instrument will be mounted on the workbench of a liquid handling robot. When loading and unloading the Stunner Plate to and from the input tray, the robotic arm has to position the Stunner Plate to a specific (X, Y, Z) point relative to the instrument. The robot must know the exact position of the instrument, in order to put the Stunner Plate in the center of the in/out tray.

To achieve this, there is a mechanical reference at the bottom side of the instrument, defining the exact position of the instrument on the workbench (Figure 6). This allows to define the (X, Y, Z) position of the Stunner Plate to put the plate on the in/out tray.

The mechanical reference for positioning the Stunner on the workbench consists of two 12 mm holes, matching 12 mm diameter pins. By placing the instrument on a plane, with the 2 pins the position of the instrument and its in/out tray is defined relative to these 2 pins. Pin specifications are:

- 10m6 (standard DIN7 pen).
- Height of pin: allowed range is 22 to 30 mm, target height is 25 mm.
- Tolerance on distance between the pins: 0.1 mm.

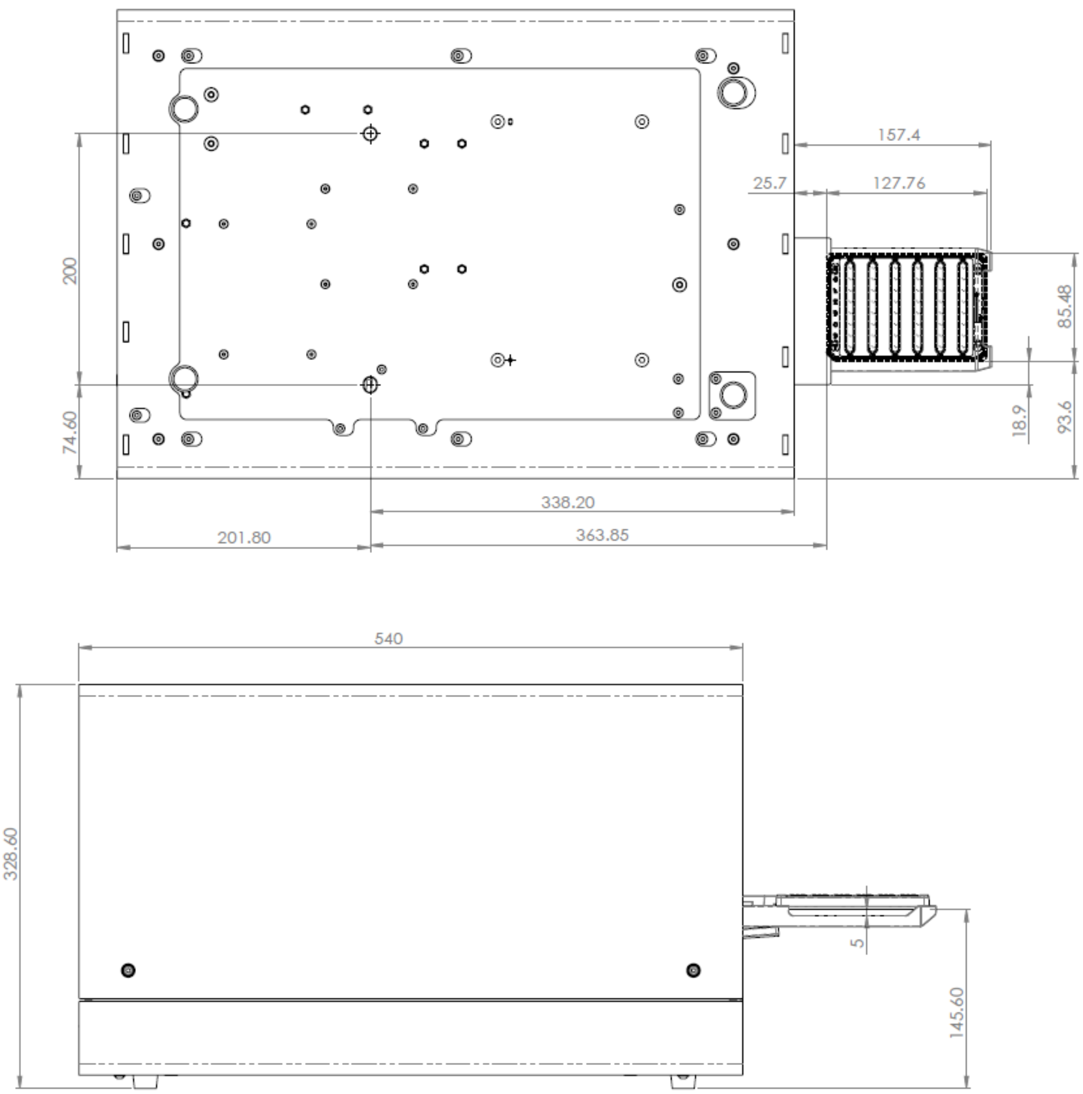



Figure 7: Features for defining the position of the in/out tray of the Stunner. The holes in the bottom plate are 12 mm diameter and act as a pre-alignment. The X, Y and Z coordinates of the Stunner Plate are defined on the figure.

When the tray is open, the Stunner Plate must be positioned at following position relative to the two pins in the workbench (Figure 7):

- Height: the ground plane of the Stunner Plate must be at 145.60 mm relative to the reference plane.
- Front position: distance from the line between the two reference pins to the projection of the middle of the Stunner Plate on the reference plane is 428.08 mm.
- Lateral position: projection of the center of the Stunner Plate on reference plane is 60.41 mm from reference pin 1 and 139.59 mm from reference pin 2.

# 3 Software integration: Setup

## 3.1 The physical setup

In the **stand-alone** setup of the instrument (also referred to as manual mode), the instrument is controlled by Lunatic & Stunner Client through a USB cable (explained in Stunner User Guide) (Figure 8).


Figure 8: The Stunner used in a stand-alone setup.

When the instrument is used in an **integrated** setup, the instrument only needs to be connected to a network by an Ethernet cable when performing the measurements (Figure 9). The only time the instrument needs to be connected to a PC running Lunatic & Stunner Client software (by a USB cable) is (a) when configuring the instrument before a first time use, (b) when upgrading or (c) when performing the calibrations.

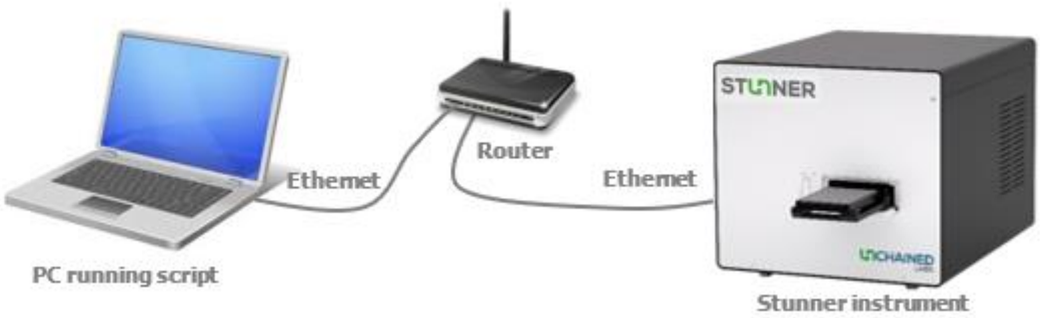



Figure 9: Stunner used in an integrated setup

To allow integration, a library with functions is provided (Stunner.dll) to control the Stunner over a remote connection. This library is called the API (Application Programming Interface). The API is typically running on a separate PC.

## 3.2 The software setup

Before using the instrument in an integrated setup for the first time, the instrument needs to be configured for remote use. The following steps need to be performed first:

1. Activating the license for API
2. Transferring the applications to the instrument
3. Configuring the network location where the experiments need to be stored

### 3.2.1 Activating the license for API

The use of the API requires a license. Please contact Unchained Labs if you do not have a license yet. The activation code necessary to activate API is based on your instruments serial number. If you request an activation code for API, please make sure you send your serial number along. Note that older systems can require a USB dongle to be plugged into the back of the instrument.

In Lunatic & Stunner Client, log in, connect to the instrument, and in the menu go to the Automation page. Enter your activation code and click “Add activation code” (Figure 10).

As an indication of the activated products a green check mark will be shown.

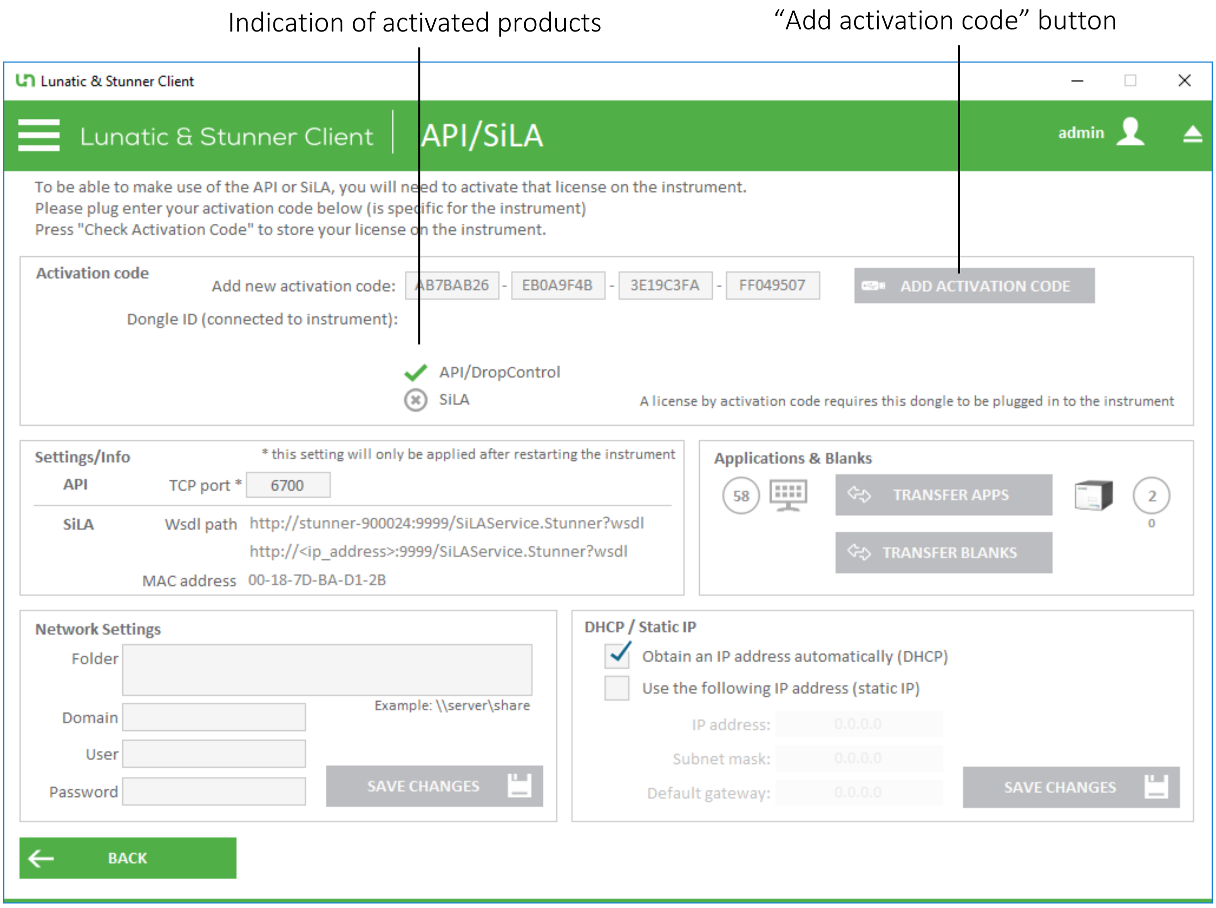



Figure 10: API license activation.

### 3.2.2 Transferring the applications to the instrument

The applications necessary to calculate the results are not installed on your instrument by default. Therefore, you need to transfer the applications to the instrument before first use and after every upgrade of the software.

In Lunatic & Stunner Client, log in, connect to the instrument, and in the menu go to the Automation page. Enter your activation code and click “Transfer apps” (Figure 11).

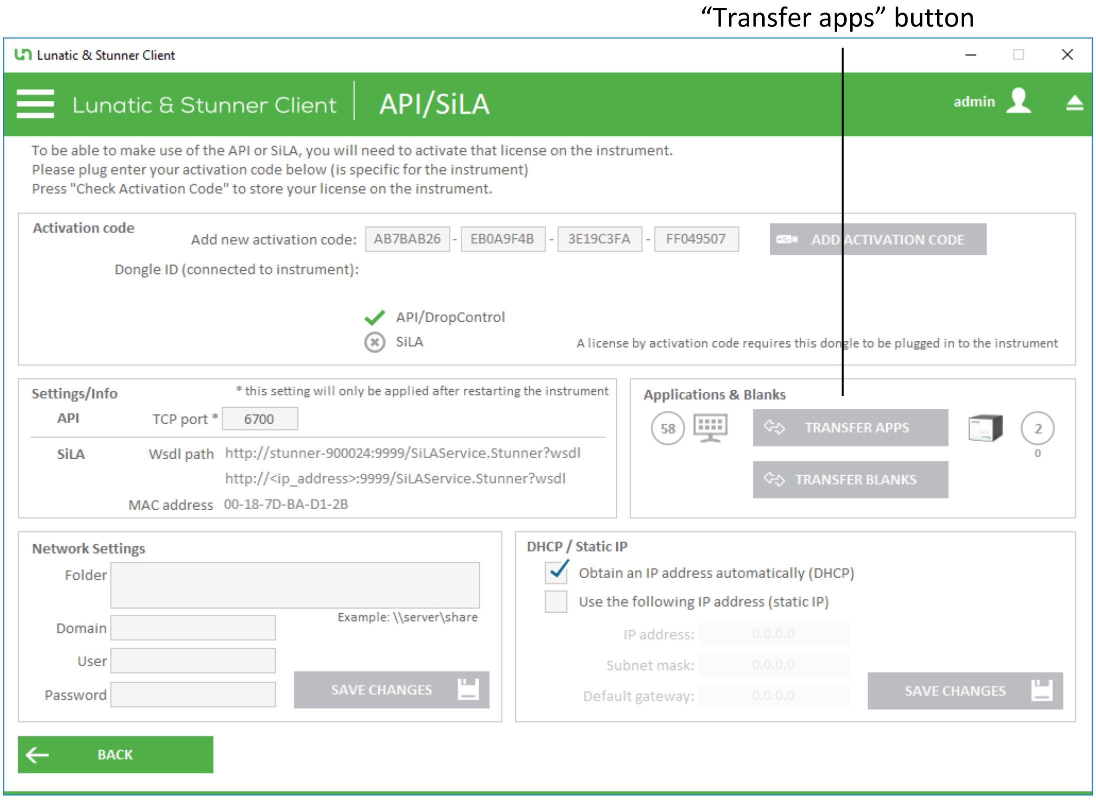



Figure 11: Applications transfer to the instrument.

### 3.2.3 Configuring the network location

The instrument needs to be configured to allow the instrument to save the experiment file to a location on the network where it has access to. Enter the network folder, domain, user and password and click "Save Changes" to transfer these settings to the instrument (Figure 12). If this folder cannot be reached, or one of the parameters is incorrect, a pop-up will be visible showing the specific error.

“Configure network folder” button

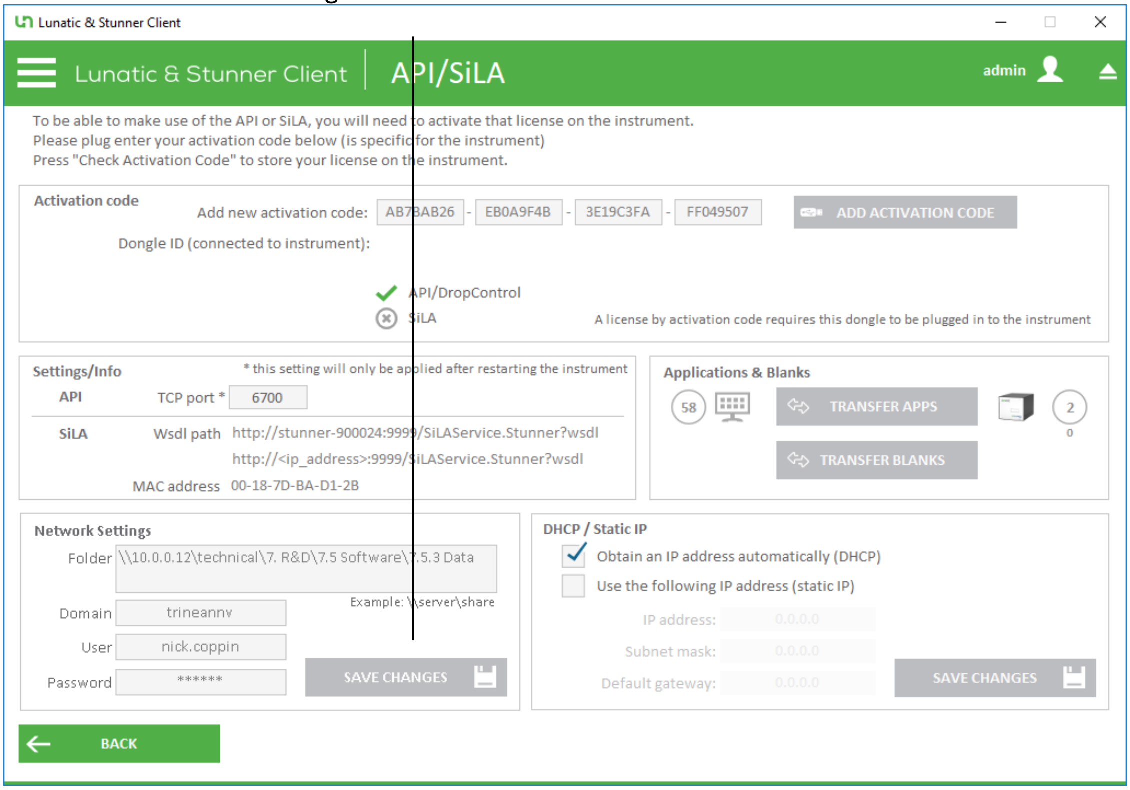



Figure 12: Configuration of the network location.

## 3.2.4 DHCP/ Static IP

The systems IP address can be assigned through Dynamic Host Configuration Protocol (DHCP) or a static IP address can be set. For Static IP, an IP address, subnet mask and Default Gateway must be entered. (Figure 13)

“IP address configuration” button

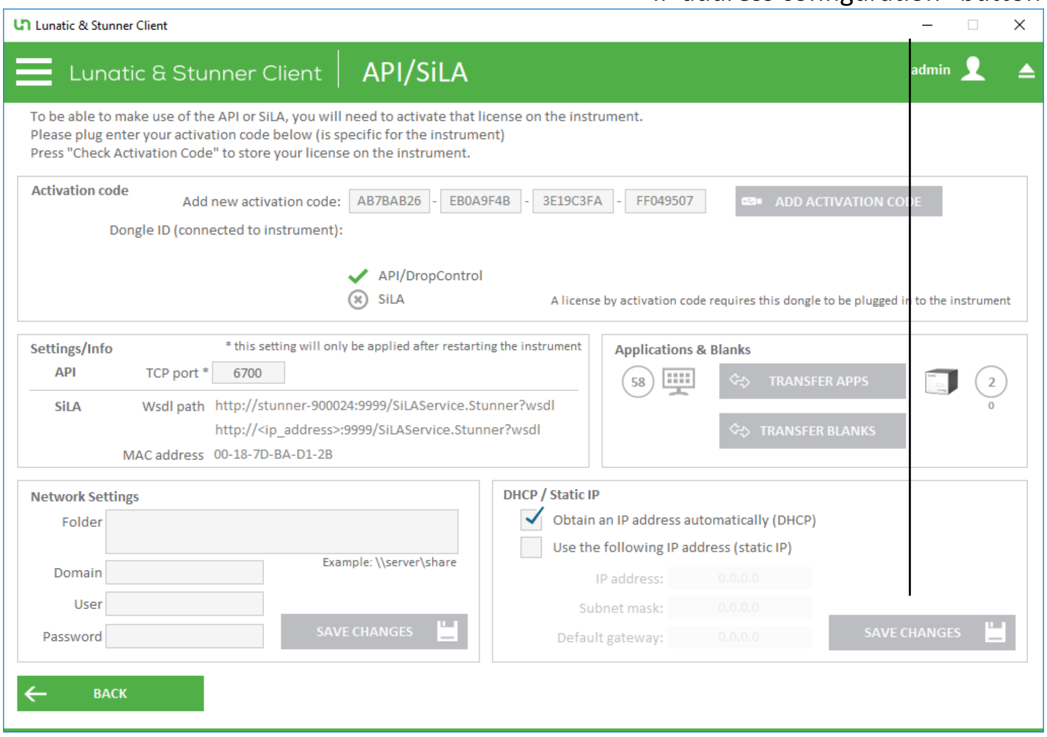



Figure 13: Configuration of the IP address.

### 3.2.5 Store blanks

Stored blanks (**Error! Reference source not found.**) allows you to reuse blanks from previous measurements. The Stunner system can store 1 experiment file for the Stunner plate type. That experiment file can consist of multiple sample groups and therefore multiple types of blanks. With the foreseen lines in experiment definition (column_stored_blanks_sample_group or stored_blanks_sample_group_name) you can specify which sample group and therefore which blank you want to use.

The file, containing the stored blanks, on the system will be overwritten each time the store Blanks command is used. The stored file will be updated for the correct plate type and the previous stored blanks will be deleted.

Blanks can be stored with 3 methods:

1. Transfer blanks viaLunatic & Stunner Client Click on the Transfer Blanks -button (Figure15) and a pop-up screen will give you an overview on all blanks that are stored and can be transferred. (Figure 16). How to store blanks in Client & Analysis software is explained in the Stunner software user guide section 3.3.
2. Use the Store blanks (**Store Blanks**) command line immediately after running an experiment with the blanks you want to store.
3. Use the Store blanks (**Store Blanks**) command line and define a file name. If you want to store blanks from an earlier experiment that is stored in your defined network folder (3.2.3).

“Transfer Blanks” button

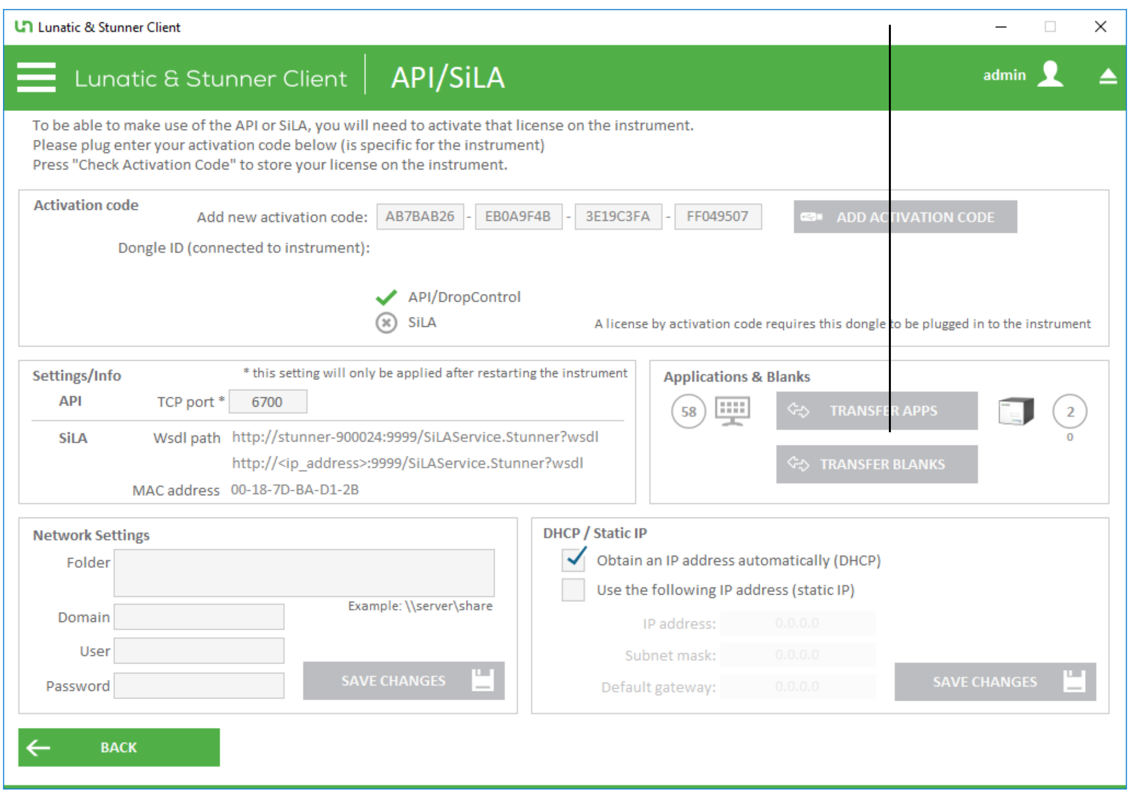


Figure 14: Transfer blanks

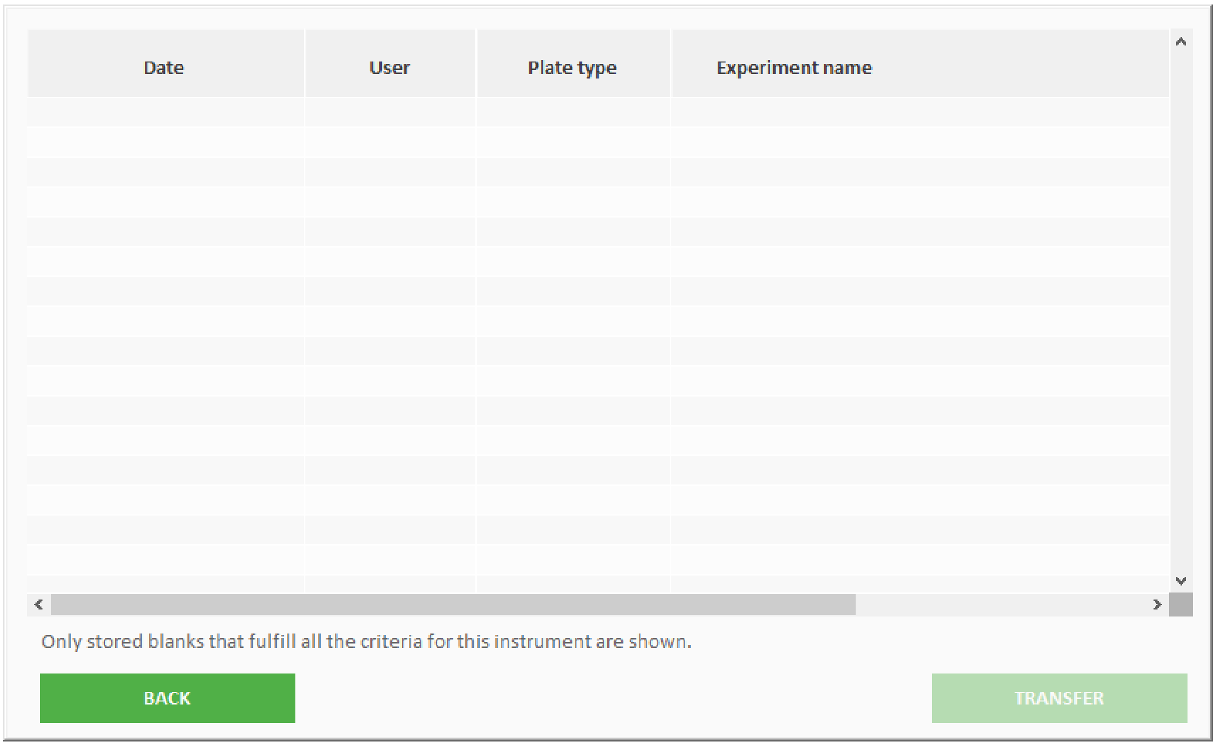


Figure 15: Transfer blanks selection screen

# 4 The programming guide

## 4.1 Structure of the script

The script is a set of functions, defining the measurement process. A measurement requires data which is organized as follows:

- There are 3 important inputs describing the experiment and the results that should be returned afterwards:
    - The **samples definition string**: name of the samples, position on the plate, etc.
    - The **experiment definition string**: Method of measurement (with or without blanks), etc.
    - The **results definition string**: format to send the results back to the lab control PC.
- The **experiment file** (*.bin) is stored in a user-defined path. This data includes all details of the measurement, including all spectra measured during pumping of the biomolecule in the measurement wells, etc.

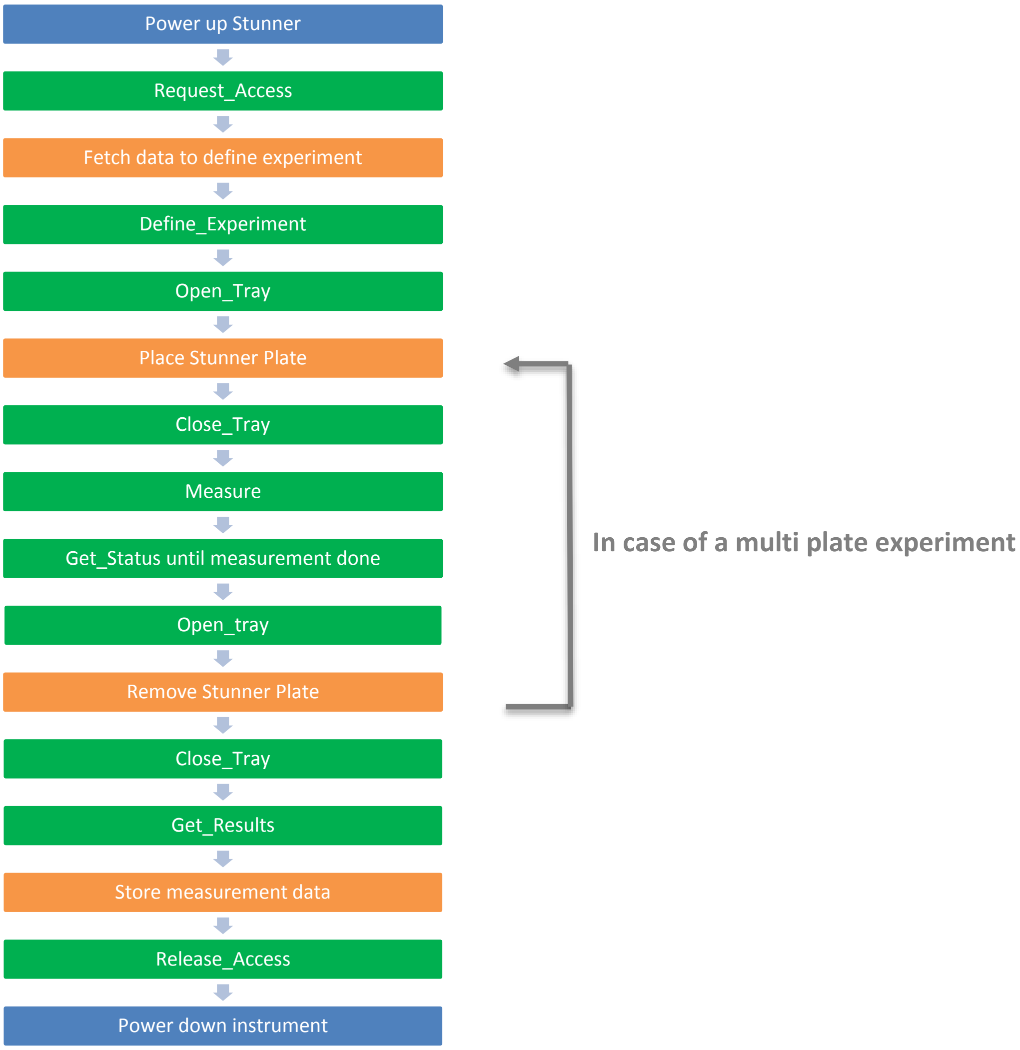



Figure 16: Example of a measurement flow. All green blocks are API commands, while the other blocks are action not done by API.

Most functions from the DLL are executed synchronously; the script waits for the results of a function call, indicating the command is executed successfully. The Measure command is an exception, here the return status shows if the command is started successfully, but it does not wait until the complete measurement was finished. The script can use the “Get Status” function to follow-up the status of the measurement and take further actions as soon as the measurement of the plates is finished (for example unloading the plate and loading a new plate).

## 4.2 Overview of the commands

This section gives an overview of all functions defined in the API to interact with the Stunner.

All functions return a response to the script, with at least the status of the system after executing the requested command. Receiving the response from the system does not always mean that the system is ready for accepting new commands. For example, with Open_Tray(), the response is sent when the tray is open (after the action). After this, the system can execute new commands. However, with Start_Measurement(), the measurement is first initialized, and a response is sent if the measurement is successfully started or not.  This way the script is not frozen during the measurement. The end of the measurement can be detected using the command Get_Status().

To minimize the number of responses, return codes are used. Appendix 2 gives an overview of these return codes. If the command was successfully executed, 0 is returned. -1 is returned if the command was not successfully executed. Potential reason of failure is found in the code, for example if you don’t have access because another remote user has access or the program is in ‘local mode’,status  -1 will be returned with “no access” as description .  . Positive numbers are used if the system returns additional information, but there’s no error. The GET_STATUS – command can be used to that additional info. E;g; when you send a Get Status command shortly after the open tray command, you will get this status return:

| | |
|---|---|
| 50 | Access and connection, tray is open |

### 4.2.1 Constructor

To create an object of the type Lunatic & Stunner Client, include the parameters ‘address’ and ‘port’ to specify the communication parameters.

Syntax1: Stunner (string address, int port)

f.e. :Stunner myStunner = new Stunner(address, port);

### 4.2.2 Request Access

Requests access to the Stunner. If no other computer has already access and the use of API is activated by a license (see Activating the license for API), access will be granted.

Syntax: myStunner. Request_Access ()

Return codes:

- status code 0: success
- Status code 1: access already active
- Status code -1: no access (instrument is already connected to another computer)
- Status code -200: could not connect over TCP

### 4.2.3 Release Access

Releases access to the Stunner, so other remote users can request access.

Syntax: Release_Access ()

Return codes:

- status code 0: success
- status code -1: no access
- status code -200: could not connect over TCP

### 4.2.4 Get Instrument S/N

Request the serial number of the instrument.

Syntax: Get_Instrument_Serial_Number ()

Return codes:

- status code 0: success
- status code -1: no access
- status code -200: could not connect over TCP

### 4.2.5 Get software version

Request the software version of the instrument.

Syntax: Get_Software_Version ()

Return codes:

- status code 0: success
- status code -1: no access
- status code -200: could not connect over TCP

### 4.2.6 Open Tray

Opens the tray if the tray is not moving.

Syntax: Open_Tray ()

Return codes:

- status code 0: success
- status code 3: tray already open
- status code -1: no access
- status code -2: Stunner instrument not in correct state
- status code -3: could not open tray
- status code -52: tray is moving
- status code -200: could not connect over TCP

### 4.2.7 Close Tray

Closes the tray.

Syntax: Close_Tray ()

Return codes:

- status code 0: success
- status code 4: tray already closed
- status code -1: no access
- status code -2: Stunner instrument not in correct state
- status code -52: tray is moving
- status code -200: could not connect over TCP

### 4.2.8 Get Status

This command can be sent at all times and will return information about access, connection to the Stunner, tray is moving/open/closed, measurement succeeded/failed/ongoing, etc. If this command is executed during a measurement, extra information can be obtained.

Syntax: Get_Status(out string measurement_info)

Return: status code (**Appendix 2: list with status code**)
+ measurement info

### 4.2.9 Define Experiment

Defines the experiment, based on 3 inputs (described in section **Definition of the interface strings**): 'ExperimentDefinition' and 'SampleDefinition'. These three inputs have specific syntax and give information about experiment name, type of measurement, material, Stunner Plate, samples/blanks, sample names, etc.

Syntax: Define_Experiment(string experimentdefinition,
string sampledefinition, out string [] plate_IDs)

Return: status code (see **Appendix 2: list with status code**)
Stunner Plate: returns a string with the list of Stunner Plate ID's that are defined by the sample definition. This list can be used to start the measurement of the Stunner Plate(s).

### 4.2.10 Start Measurement

Starts the measurement, based on the Plate_ID. Every Stunner Plate measurement needs to be started by a command. To know when the previous measurement is finished, use Get Status.

Syntax: Measure(string Plate_ID)

Return: status code (see **Appendix 2: list with status code**)

Hint: if you want to use the internal barcode scanner, use the keyword "autodetect" as Plate_ID!

### 4.2.11 Abort Measurement

Stops the measurement. The ongoing measurement will be interrupted by this command.

Syntax: Abort_Measurement ()

Return: status code (see **Appendix 2: list with status code**)

### 4.2.12 Get Results

Returns the results of the measurement, 'ResultParameters' (described in **Definition of the interface strings**) determines what columns are in the report, what is used as separator and where the results are stored. If you enter a Plate_ID, only the results of that Stunner Plate will be given, if you leave it blank, all results will be given.

Syntax: Get_Results(string results_parameters, string plate_ID, out string results, out string experiment_path)

### 4.2.13 Read bar code

Reads the bar code of the Stunner Plate (already in the system).

Syntax: Read_Bar_Code(out string bar_code)

or

Read_Bar_Code(out string bar_code, out bool valid, out string plate_type)

Return: status code (see **Appendix 2: list with status code**)

### 4.2.14 Pause

Pauses all actions, until command Continue is sent.

Syntax: Pause ()

Return: status code (see **Appendix 2: list with status code**)

### 4.2.15 Continue

Resumes the measurement that was paused.

Syntax: Do_Continue ()

Return: status code (see **Appendix 2: list with status code**)

### 4.2.16 Reset

Gets the instrument out of error state.

Syntax: Reset ()

Return: status code (see **Appendix 2: list with status code**)

### 4.2.17 Store Blanks

This command can be sent after running an experiment containing blanks you want to store.
If you want to store blanks from an earlier experiment that is stored in your defined network folder (3.2.3), you need to define the experiment file name. Only 1 experiment canbe stored on the instrument. However, this experiment can contain multiple sample/blank groups. This command will overwrite the previously stored blanks if the new experiment passes the needed criteria.

The blanks need to pass the following criteria to be successfully stored:

- Minimal 3 successful replicates that pass a reproducibility check
- Instrument has not been calibrated since the blank measurement
- Blank measurement is not more than 100 days old
- Blank measurement is done on the same instrument

Syntax: Store_Blanks () or Store_Blanks (string file_name)

Return: status code (see **Appendix 2: list with status code**)

### 4.2.18 Get Stored Blanks

Returns a list of stored blanks on the instrument.

Syntax: Get_Stored_Blanks()

Return: status code (see **Appendix 2: list with status code**)
Returns a string with information on Stored blank:
- Plate type
- Sample group names
- Number of days still valid

### 4.2.19 Debugging

To debug the system, there are different options: check the return code and use the GetLastInternalError() function.

The return code is generated by each DLL function, and contains information on the success of the function call. When you get a return code that is not zero, you can check "**Appendix 2: list with status code**" to get the description of the error that occurred. Most of the times the error codes deliver an obvious solution, f.e. -2: Not connected to the Stunner. Others require a bit more research, f.e. -9: Error reading experiment definition.

To allow debugging when a problem is encountered in the API function itself, the following extra function is created to extract further information on the occurred error:

Syntax: GetLastInternalError ()

Return: string with a more detailed description of what went wrong inside the DLL

Note: this function will only return an error string when the previous command returned a -2xx error code (-200, -201 or -202).

It is recommended to use this function in the customer script, to catch error messages under all circumstances.

## 4.3 Definition of the interface strings

To run a measurement, the instrument needs crucial information such as the number of samples, required results export format, etc. This is done via a number of text strings. These are:

- Sample Definition String: text file or string describing the samples: position, name, reference to the Stunner Plate, reference to the source plate, etc.
- Experiment Definition String: text file defining the settings of the experiment (such as name of the experiment, Stunner Plate type, etc)
- Results Definition String: text file defining the format of the results file.

The **Sample Definition String** is a text file with the definition of the samples. It is a table with columns and rows. Each row represents one sample, and in the columns the Stunner Plate name, position, sample name etc. of the sample are defined. The columns may be in a different order, as long as the Experiment Definition String is configured accordingly (see next section).

An example of a sample definition string is shown in table 2:

```
Plate 1,A1,blank1,Plate 1,A1,SG1,BSA,PBS
Plate 1,B1,sample1,Plate 1,A1,SG1,BSA,PBS,6
Plate 1,C1,sample2,Plate 1,A1,SG1,BSA,PBS,7
Plate 1,D1,blank2,Plate 1,D1,SG2,IgG,water
Plate 1,E1,sample3,Plate 1,D1,SG2,IgG,water,13.7
Plate 1,F1,sample4,Plate 1,D1,SG2,IgG,water,12.1
```

Table 2: Example Sample Definition String with 4 samples on one plate, 2l samples using A1 as blank reference, 2 samples using D1 as blank reference..

Note an experiment may contain multiple plates: all samples of the plates defined in the experiment are collected in one (large) sample definition string. After definition of the experiment, an array with the Plate_ID's defined in the sample definition string is returned. If the script tries to measure more plates than defined in the sample definition string, or if the script asks the Stunner to send the results without having measured all plates defined in the Sample Definition File, then an error message is returned.

It is the responsibility of the script programmer to offer the exact number of plates to the Stunner as defined in the sample definition string.
After defining an experiment, the Stunner returns an array with names of the plates defined (cfr paragraph **Define Experiment**).

**Experiment Definition String** contains 2 sections: [Experiment definition] and [Import samples]. The [Experiment definition] section describes the general settings of the experiment. The [Import samples] section describes the format of the sample definition file. Following settings must be in the [Experiment definition] section:

- experiment_name: user defined name for the experiment[1]
- application_name: use the application name as it is visible in the Lunatic & Stunner Client on the Select app screen
- plate_type: has to be "Stunner Plate"

The [Import samples] section contains next fields:

- column_source_plate: column number in the Samples definition string, containing the source plate ID (this is a reference to the plate in the biobank in which the extracted DNA is stored for example, typically this is a string with the plate name)
- column_source_position: column number in the Samples definition string containing the position (in A1 .. H12 notation)
- column_plate_ID: column number containing the ID of the Plate
- column_plate_position: column number containing the postion on the Plate (in A1..H12 notation)
- column_sample_name= column number containing the sample name
- column_blank_plate_ID= column number containing the ID of the frame with blank
- column_blank_plate_position= column number of the position of the blank.
- column_sample_group= column number containing the defined sample group.
- column_analyte= column number containing the defined analyte.
- column_buffer= column number containing the defined buffer.
- column_E1%= column number containing the extinction coefficient of the sample.
- column_stored_blanks_sample_group= column number of the sample group in case you are using stored blanks with multiple sample groups
- dls_acquisition_time= the time of a single acquisition in seconds
- dls_number_of_acquisitions= column number containing the preferred amount of acquisitions.
- blanking_information sets the way the blank sample is defined:
  - <u>0 = autoblank</u>: no blank measurement is done. Note this setting is not recommended for accurate measurements of low concentrated samples.
  - <u>1 = averaging of blanks</u>: in that case, the field blank_name_used contains the name of the blank sample. This name must be defined in the Samples definition string (and

[1] If the string contains whitespaces, then add a quotation (") before and after the string.

must be defined multiple times in the Samples definition file, so that the instrument can calculate the average of these measurements)
    - 2 = single blank: the blank is defined in the field “blank_name_used” (where the name must be defined in the Samples definition string), or in the combination of the field “blank_plate_ID” and “blank_plate_position”.
    - 3 = multiple blanks: the blank for each sample is defined by the fields “blank_droppplate_ID” and “blank_plate_position”
    - 4 = stored blanks: Use the blanks stored on the instrument. In case of multiple sample groups, use the field “stored_blanks_sample_group_name” or “column_stored_blanks_sample_group”
- blank_plate_ID = the Plate ID of the blank in case of one blank for the complete experiment (see definition of the field blanking information). May be empty in other cases.
- blank_plate_position = the blank position (A1 notation) (see definition of the field blanking information). May be empty in other cases.
- blank_name_used = name of the blank (as defined in the Samples definition file). (see definition of the field blanking information)
- stored_blanks_sample_group_name= name of the stored blank sample group you want to blank with. In case your stored blank experiment contained multiple sample groups.

An example is shown in table 3:

```
[Experiment definition]
experiment_name="Experiment by remote"
application_name="Sizing & polydispersity"
plate_type="Stunner Plate"

[Import samples]
column_source_plate=-1
column_source_position=-1
column_plate_ID=0
column_plate_position=1
column_sample_name=2
column_blank_plate_ID=3
column_blank_plate_position=4
column_sample_group=5
column_analyte=6
column_buffer=7
column_E1%=8
blanking_information=3
blank_plate_ID=""
blank_plate_position=""
blank_name_used=""
stored_blanks_sample_group_name=""
column_stored_blanks_sample_group_name=-1
```

Table 3: Example of experiment definition string.

The **Results Definition String** is a text file defining the output format for the results. These are stored in a table-based text file (Table 4).

```
[Export results]
column_names="Plate ID;Plate Position;Sample name; Sample group
name,Pump;Analyte;Buffer;Concentration (mg/ml);A280 (10mm);Peak 1 Mean Dia (nm)"
separator=;
undefined_column_name="remove"
no_result_value="-"
```

Table 4: Example of results export format file.

- column_names: defines which columns to be included in the results. The name of the columns is exactly the same as defined in the export window in Lunatic & Stunner Analysis.

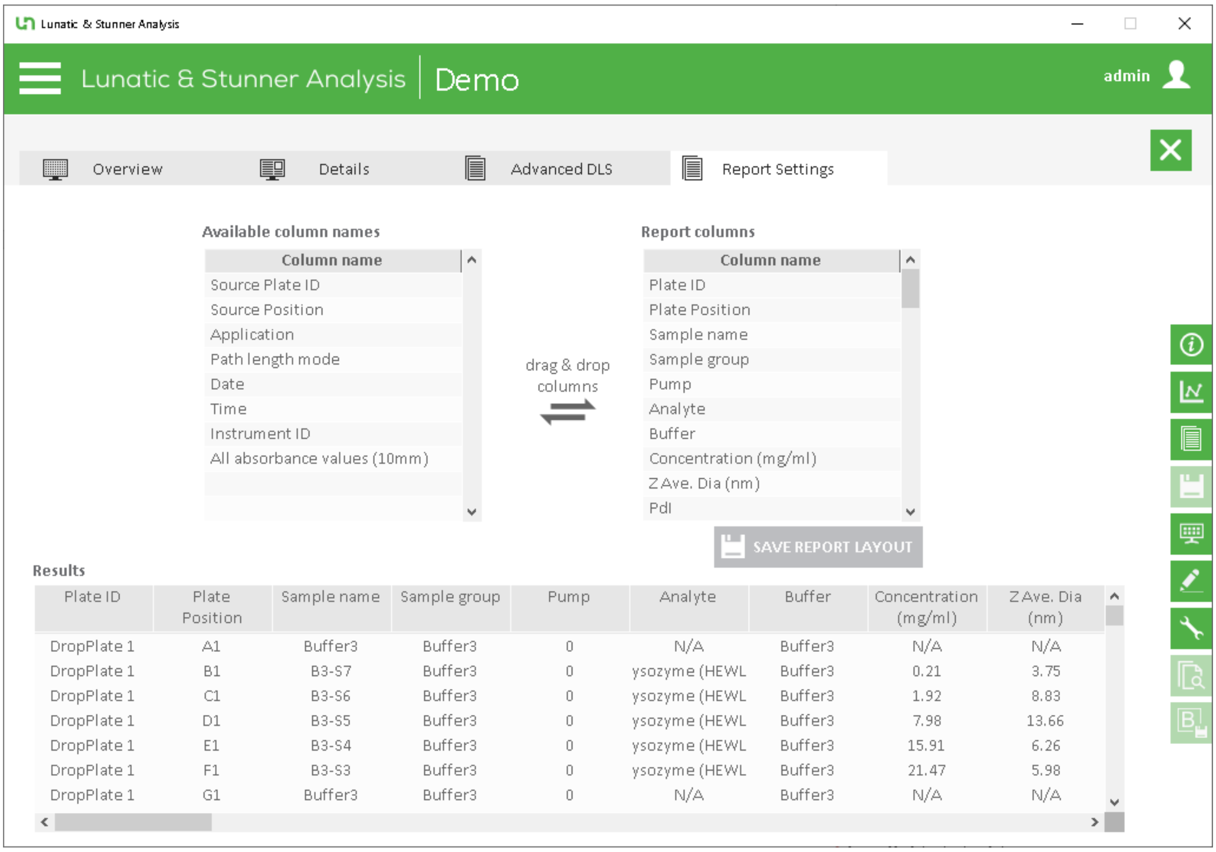


Figure 17: The export window (from standard user interface) showing the allowed column names available for export.

- separator: determines the character that will separate the different columns in the results (values that are allowed are “;” “,” and “tab”).
- undefined_column_name= defines what needs to be done when a column is requested that does not exist (values that are allowed are "remove",”include”, Return_error”).
- no_result_value: this value will be returned when no result is present (e.g. unmix results for blanks or concentrations of positions that did not contain any fluid). Default value is “-“, but can be ””, “N/A”, “0”, …

The instrument returns a text file with contents as defined in the Results Definition String.

# 5 Appendix 1: example

A simple example of the script is shown in next table (in C#)

```
using System;
using System.Collections.Generic;
using System.Text;
using Stunner;

class Program
{
   static void Main(string[] args)
   {
      string address = "192.168.0.100";   //set the IP address
      int port = 6300;               // set the TCP port
      // add the experiment definition or read it from a file
      string exp_def =@"
[Experiment definition]
experiment_name=""Experiment by remote""
application_name=""Sizing & polydispersity""
plate_type=”Stunner Plate”

[Import samples]
column_source_plate=-1
column_source_position=-1
column_plate_ID=0
column_plate_position=1
column_sample_name=2
column_blank_plate_ID=3
column_blank_plate_position=4
column_sample_group=5
column_analyte=6
column_buffer=7
column_E1%=8
blanking_information=3
blank_plate_ID=""""
blank_plate_position="""""";
      // add the samples or read them from a file
      string samples = @"Plate 1,A1,blank1,Plate 1,A1,SG1,BSA,PBS
Plate 1,B1,sample1,Plate 1,A1,SG1,BSA,PBS,6
Plate 1,C1,sample2,Plate 1,A1,SG1,BSA,PBS,7
Plate 1,D1,blank2,Plate 1,D1,SG2,IgG,water
Plate 1,E1,sample3,Plate 1,D1,SG2,IgG,water,13.7
Plate 1,F1,sample4,Plate 1,D1,SG2,IgG,water,12.1";
      // add the result parameters or read them from a file
      string result_parameters = @"
[Export results]
column_names=""Plate ID;Plate Position;Sample name; Sample group,Pump;Analyte;Buffer;Concentration
(mg/ml);A280 (10mm);Peak 1 Mean Dia (nm)""
separator=;
undefined_column_name=""remove""
no_result_value=""-""
      string [] plates;
      int status;
      // create Stunner object
      Stunner myStunner = new Stunner(address, port);
```

```
        if (myStunner.Request_Access() < 0) { goto Exit; }
        if (myStunner.Define_Experiment(exp_def, samples, out plates) < 0)
                        { goto End; }
        for (int i = 0; i < plates.Length; i++)
        {
            if (myStunner.Open_Tray() < 0) { goto End; }
            // load the Plate
            if (myStunner.Close_Tray() < 0) { goto End; }
            if (myStunner.Measure(plates[i]) < 0) { goto End; }
            do
            {
                status = myStunner.Get_Status();
                if (status != 32 & status != 25 & status != 31)
                            { goto End; }
            }
            while (status != 32 & status != 25);
        }
        string results, experiment_path;
        do
        {
            status = myStunner.Get_Results(result_parameters, "", out results,
                                                                        out experiment_path);
            if (status == 0)
            {
                //write results to the location you want
                break;
            }
        } while (status == -104);
                // -104 is send when not all plates are processed yet
    End:
        status = myStunner.Release_Access();
    Exit:
    }
}
```

Next table is an example of the output file in text format:

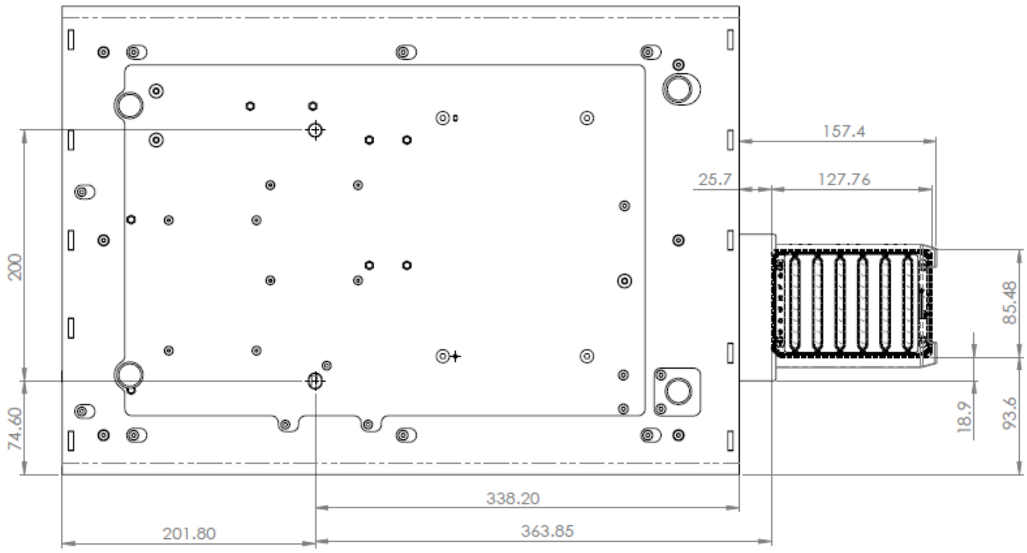
```
Plate ID;Plate Position;Sample name;Sample
group,Pump;Analyte;Buffer;Concentration (mg/ml);Peak 1 Mean Dia (nm)
Plate 1;A1;blank1;Plate 1;A1;SG1;0;-;PBS;-;-
Plate 1;B1;sample1;Plate 1;A1;SG1;0;BSA;PBS;2.36;8.78
Plate 1;C1;sample2;Plate 1;A1;SG1;0;BSA;PBS;2.44;8.57
Plate 1;D1;blank2;Plate 1;D1;SG2;0;-;water;-;-
Plate 1;E1;sample3;Plate 1;D1;SG2;0;IgG;water;14.57;7.89
Plate 1;F1;sample4;Plate 1;D1;SG2;0;IgG;water;7.24;8.7
```

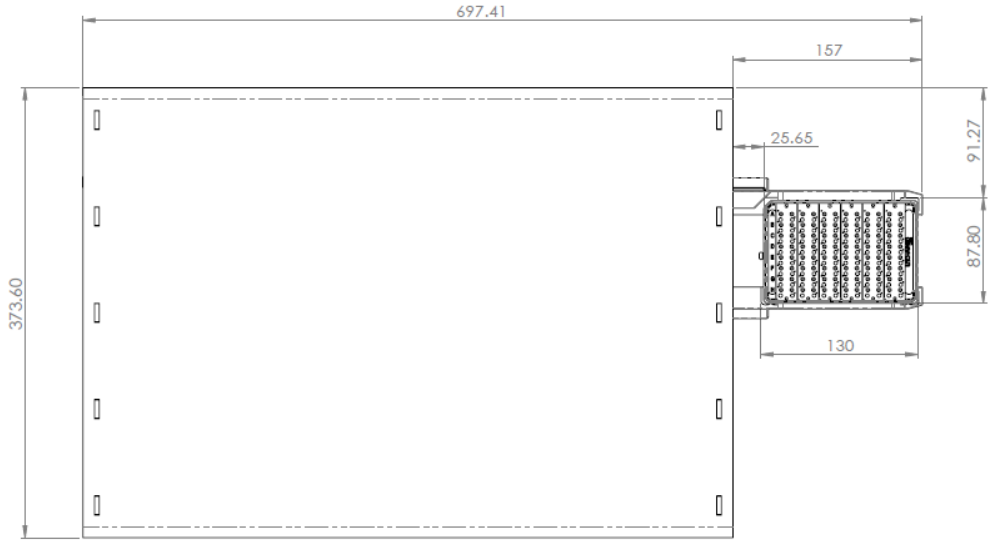
# 6 Appendix 2: list with status code

| | |
|---|---|
| -904 | Plate type cannot be used with this instrument |
| -903 | Plate type cannot be used with this application |
| -902 | Unknown plate type in experiment definition |
| -901 | Application not installed on instrument |
| -401 | File listed as an argument could not be found/read |
| -400 | Not all arguments are listed |
| -300 | Unknown command |
| -202 | Internal DLL error: Could not convert the status code to an integer |
| -201 | Internal DLL error: Could not convert the DQ return string to the desired parameters |
| -200 | Could not connect over TCP |
| -120 | Automation license not activated on instrument |
| -111 | Instrument in incorrect state |
| -106 | No Plate in the instrument |
| -104 | Results can only be exported if all Lunatic Plates are measured |
| -103 | Incorrect argument used, Lunatic Plate ID is already measured |
| -102 | Incorrect argument used, Lunatic Plate ID not valid |
| -100 | Access interrupted by local user interface |
| -99 | Timeout |
| -68 | Store blanks: More than one stored blank sample group available |
| -67 | Store blanks: Stored blank sample group name incorrect (by name) |
| -66 | Store blanks: Stored blank sample group names incorrect (by column) |
| -65 | Store blanks: File not found |
| -64 | Store blanks: No reproducible blanks found |
| -63 | Store blanks: Instrument calibrated since blank experiment |
| -62 | Store blanks: Blank experiment too long ago |
| -61 | Store blanks: Blank experiment is from a different instrument |
| -52 | Access and connection, could not perform command, tray is moving |
| -31 | Could not perform command, measurement busy |
| -24 | Access and connection, measurement failed |
| -22 | Access and connection, measurement failed to initialize |
| -17 | Bar code reader timeout |
| -16 | No bar code reader available |
| -15 | Bar code not valid |
| -14 | Timeout for going to open tray position |
| -13 | No samples defined |
| -12 | No pump profiles found |
| -11 | No pump profiles found |
| -10 | Error reading samples |
| -9 | Error reading experiment definition |
| -5 | Tray not in right position |
| -4 | Could not close tray |
| -3 | Could not open tray |
| -2 | Instrument in incorrect state |
| -1 | No access |
| 0 | Successful |
| 1 | Access already achieved |
| 3 | Tray was already open |

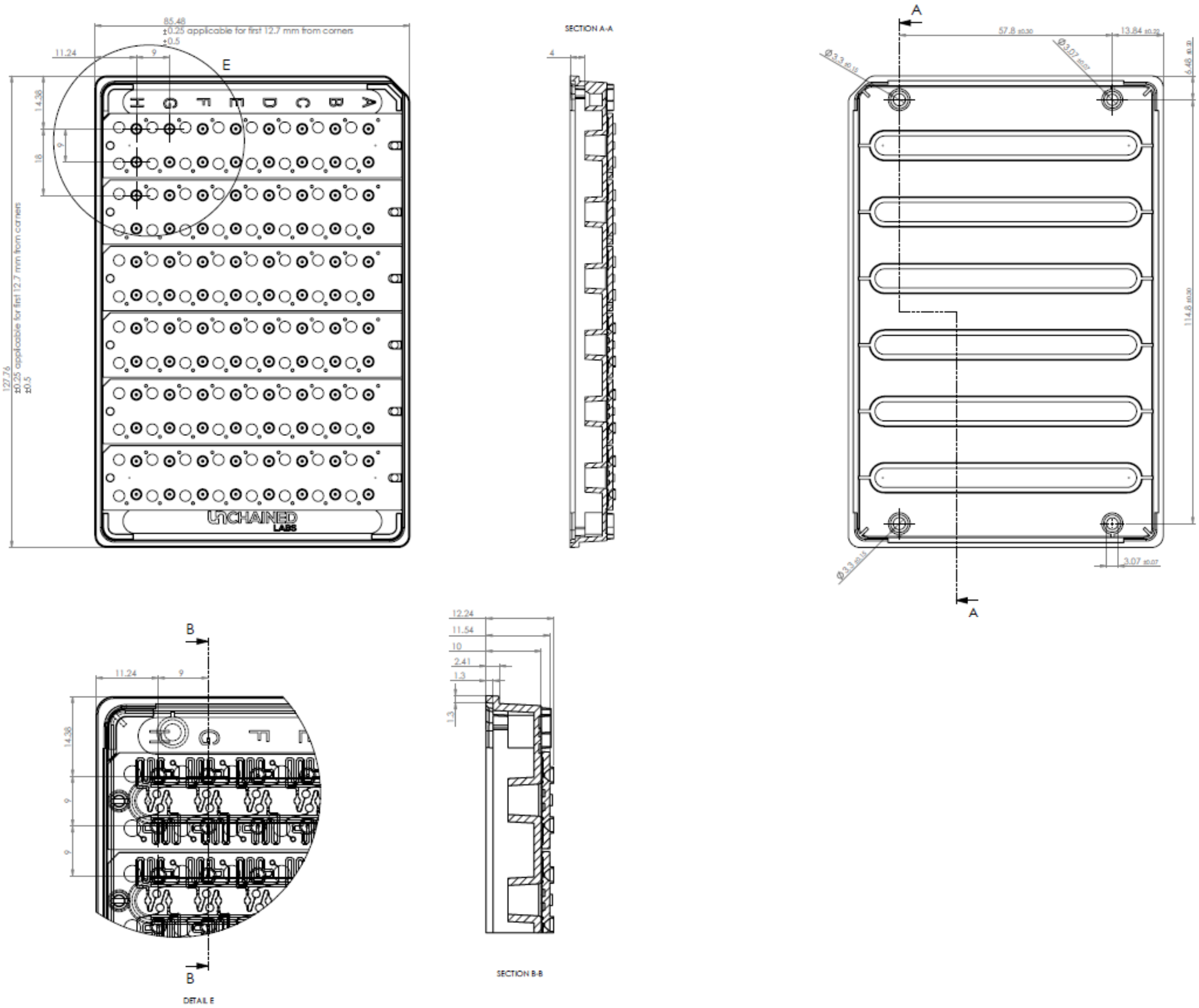
| | |
|---|---|
| 4 | Tray was already closed |
| 20 | Access free |
| 21 | Access and connection, no measurement |
| 23 | Access and connection, measurement started |
| 25 | Access and connection, measurement successful |
| 30 | Access and connection, measurement initializing |
| 31 | Access and connection, measurement busy |
| 32 | Access and connection, load next Lunatic Plate |
| 33 | Access and connection, measurement paused |
| 50 | Access and connection, tray is open |
| 51 | Access and connection, tray is closed |
| 52 | Access and connection, tray is moving |
| 999 | Lunatic & Stunner Client Exit accepted |

# 7 Appendix 3: the outer dimensions of the instrument





697.41
157
25.65
91.27
87.80
373.60
130

# 8 Appendix 4: The dimensions of the plates



6870 Koll Center Parkway
Pleasanton, CA 94566
**Phone:** 1.925.587.9800
**Toll free:** 1.800.815.6384
**Email:** info@unchainedlabs.com