Phasespace Guide

Getting Started

Step 0: Look through the documentation

The Phasespace company locks down the documentation, so it is not freely available on the internet.

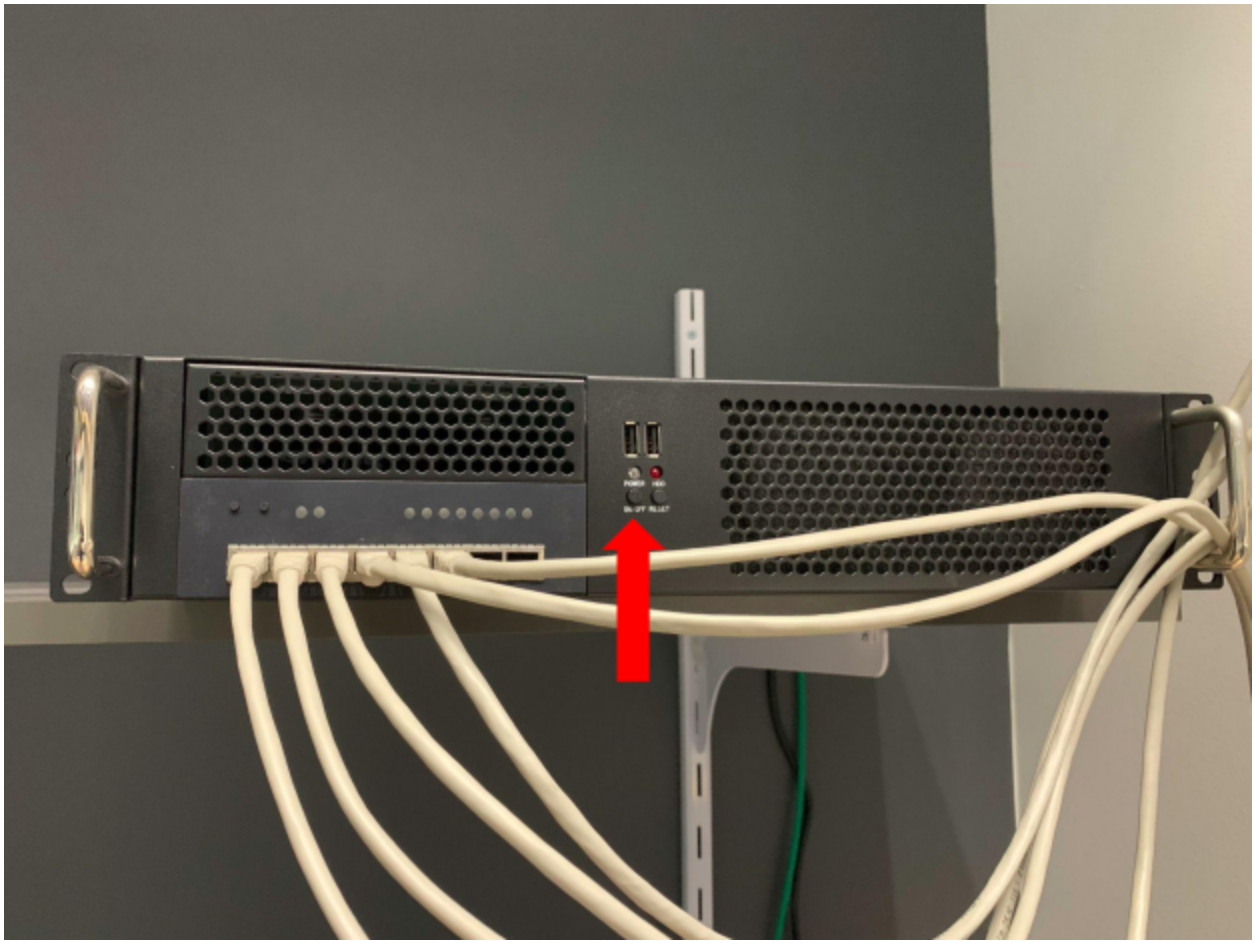
You can find official documentation at the link below:

https://customers.phasespace.com/anonymous/ImpulseX2E/
username: anonymous
password: guest

You will be able to find the Quick Start guide, the Master Client guide, and the C++ API guide here.

Step 1: Turn on the Phasespace hub

The hub is a short black box located near the entrance of the Drone Lab. Turn it on by pressing the “on” button on the front of the box.



Above: A picture of the front of the Phasespace hub. The on/off button is on the left. As pictured, the hub is off, as neither LED is turned on.

Step 2: Connect to client

You may use either the Configuration Manager (a web application) or the Master Client (a Windows-only desktop application) to interact with the phasespace.

Using the Configuration Manager

For our Phasespace system, you can access the configuration manager at:

http://cs-phasespace.cs.umn.edu
username: admin
password: phasespace

Calibration + Alignment

The Phasespace system needs to be calibrated from time to time. For example, each time the net is lowered/raised, the cameras might move slightly, making it necessary to recalibrate the system. This is done by waving a calibration wand around the space until calibration data is generated for each camera.

Step 1: Connecting the micro-driver into the wand.

The wand is typically located in one of the cabinets that are on the same wall as the Phasespace system.

Above: A picture of the calibration wand

Remove the wand from the cabinet, as well as the microdriver. The microdriver is a small grey rectangular object with a port on one of the short ends.

Above: A picture of the Phasespace microdriver

Plug the microdriver into the wand. The notch in the middle of the large port is off center, so it will only plug into the wand one way.

Step 2: Turning on the microdriver

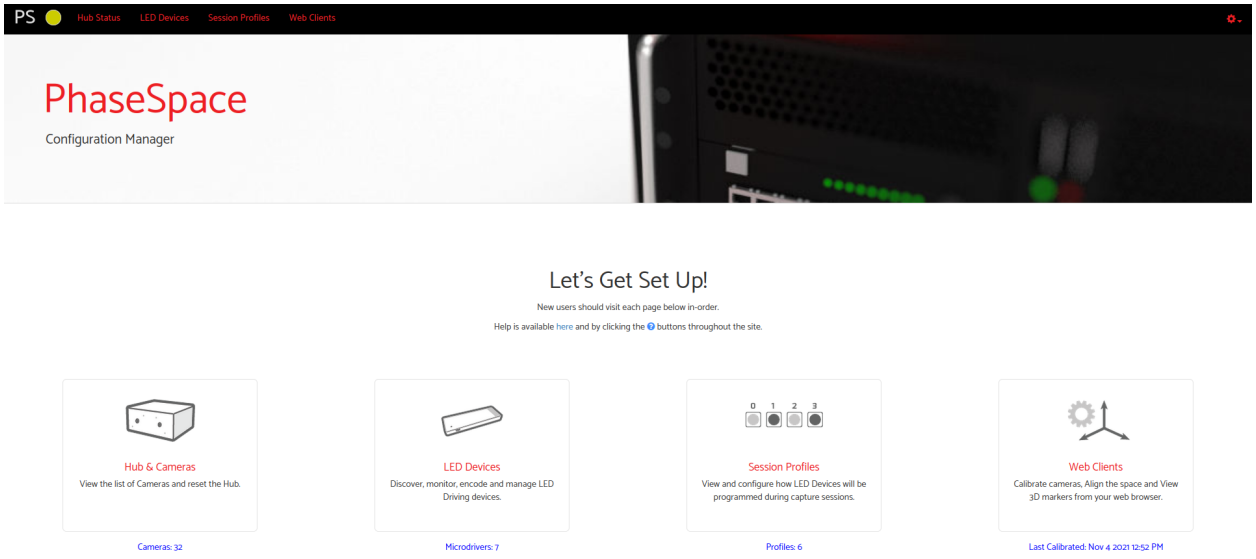
The microdriver controls the active marker LEDs (it provides power + it connects to the phasespace hub).

Turn it on by pressing and holding the white button until the orange LED turns on

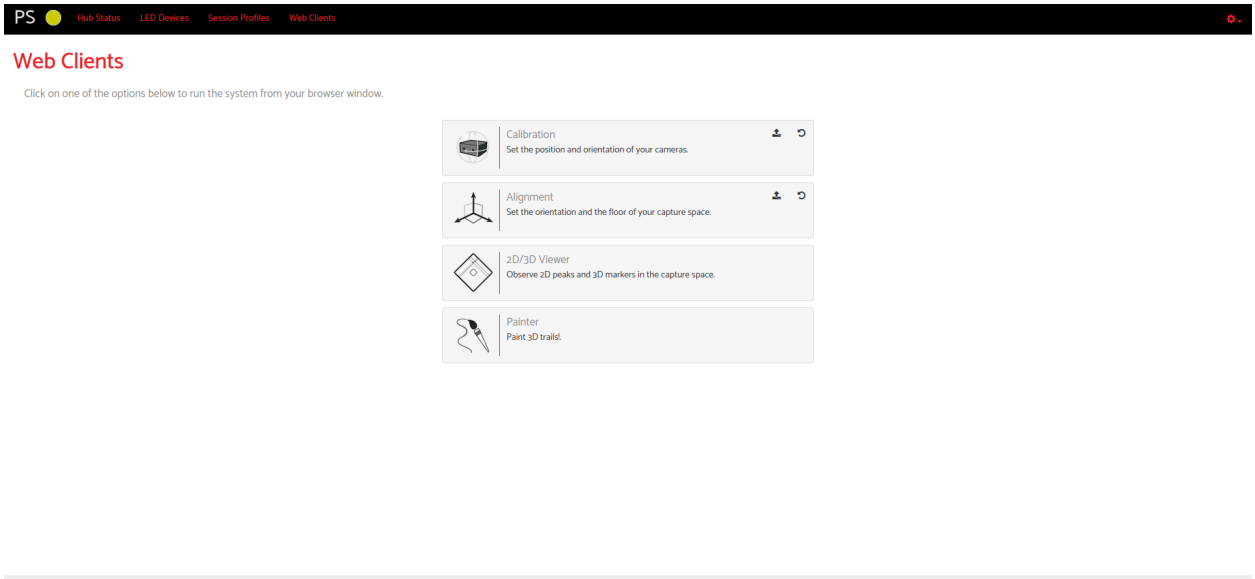
[add picture of button, led]

Step 3 (calibration): Use the calibration client in the Configuration Manager to calibrate the system

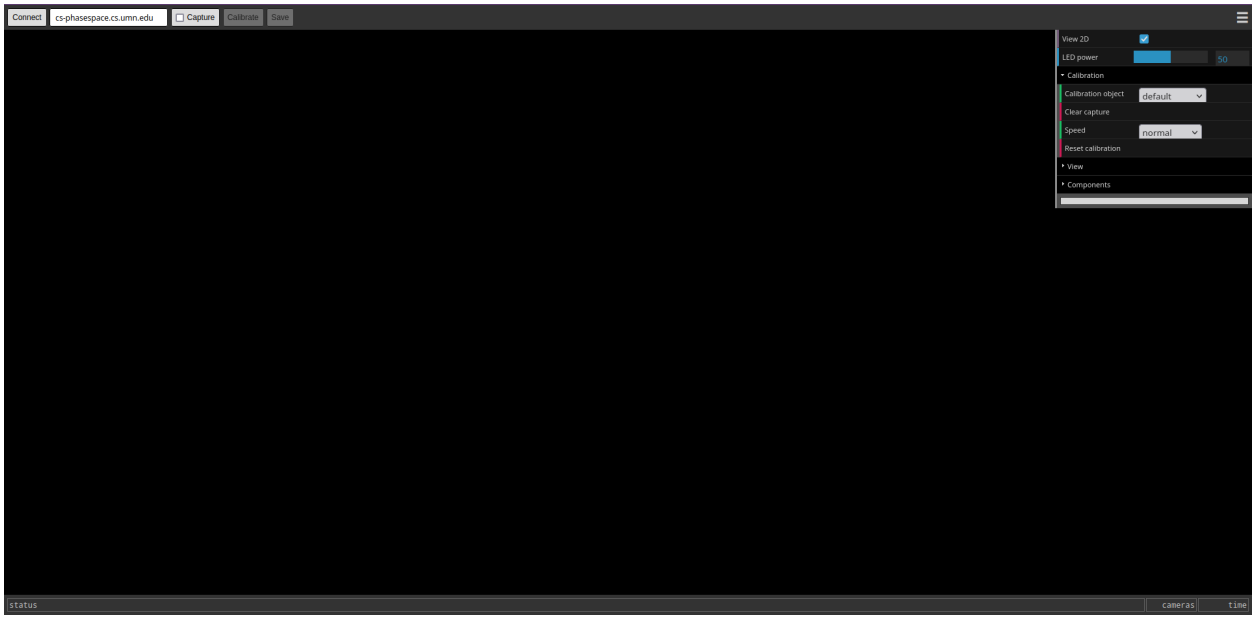
Connect to the Configuration Manager as detailed above. Click on “Web Clients”, then “Calibration”.



Above: landing page for Phasespace Configuration Manager



Above: Web Clients page



Above: Calibration client

At the top left, click the “Connect” button. Once connected, the LEDs on the calibration wand should turn on. Additionally, the screen should fill with many squares. Each square represents a camera.

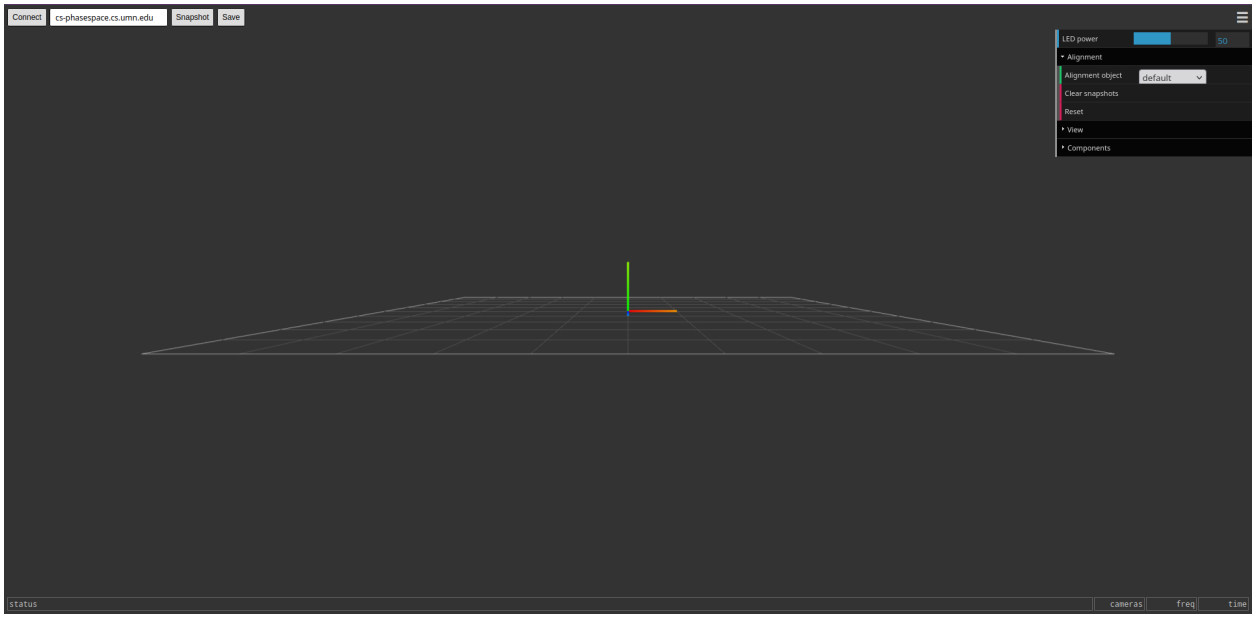
When you are ready to start moving the wand around the space, click the “Capture” checkbox. Proceed to move the wand around the room. As you pass the wand through a camera’s field of view, its corresponding square in the calibration client should progressively turn green. Continue moving the wand around the space until all of the squares are at least 50% green.

[add picture of calibration where all of the squares are at least 50% green]

Once you have reached this point, you can click the “calibrate” and “save” buttons
Click the “Disconnect” button, then close the tab. The phasespace system has been calibrated.

Step 4 (alignment): Use the alignment client in the Configuration Manager

Now that the phasespace system is calibrated, we need to set the origin for the world frame. First, open the alignment client through the Web Clients selection page. Similarly to the calibration client, this will open a new window. Click the connect button in the top left corner.



Move the calibration wand to where you want the origin to be, and hold it upright. There should be a piece of tape on the ground suggesting an origin.

[picture of where the origin is + wand upright]

While the alignment wand is being held upright at the origin, click the “snapshot” button. You may need to have someone else click the button.

[picture of the snapshot button]

Proceed to take a step in the +x direction, and click the snapshot button again. Then, take a step back, and take a step in the +z direction (to the right of the +x direction; the phasespace has the y axis pointing up).

[2 pictures of the +x and +z direction]

If you are satisfied with your results, click “save” and then “disconnect”. Otherwise, click the “reset” button on the right hand side.

[picture of the save, disconnect, and reset buttons]

Using motion capture data from the Phasespace system

There are two ways to use motion capture data from the Phasespace system: in real time using the C++ SDK, or in an offline way by recording the data.

Note that the C++ SDK can be downloaded from here:

https://customers.phasespace.com/anonymous/SDK/5.2/
username: anonymous
password: guest

and the Master Client can be downloaded from here:

https://customers.phasespace.com/anonymous/Software/
username: anonymous
password: guest

Using the C++ SDK with ROS

The Phasespace C++ SDK is relatively simple -- a single shared object file accompanied by a couple of header files.

The example ROS node provides a good starting point for writing your own code that interfaces with the phasespace system. The example ROS node has been verified to work on Ubuntu 16.04 + ROS Kinetic.

The Phasespace SDK does not have any special considerations for ROS (e.g unlike the DJI SDK, which has special facilities for using ROS along with it), so it is straightforward to take code from the non-ROS examples (such as the example code for tracking the position and orientation of rigid bodies, as opposed to just the position of individual marker points).

Recording data with the Master Client

The master client is a Windows-only application. However, it seems to work well under WINE, so that is likely a viable option for Phasespace users who have a Linux or macOS computer.

TODO

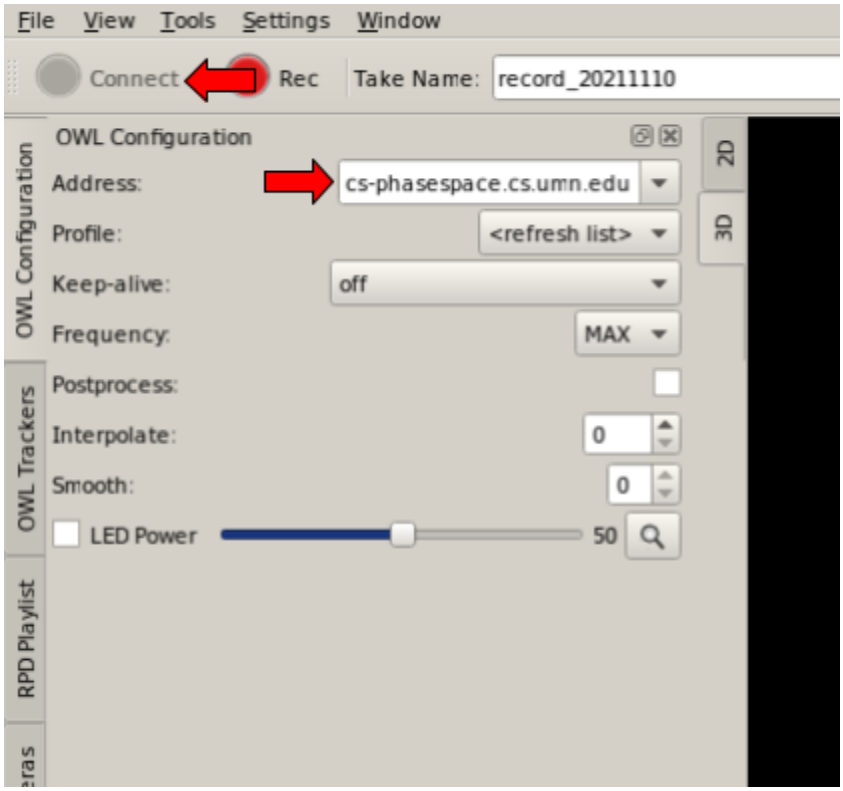
- What kinds of files will the Master Client produce for a recording?
- What tools are there to interact with those files?

Tracking rigid bodies with the Phasespace system SDK

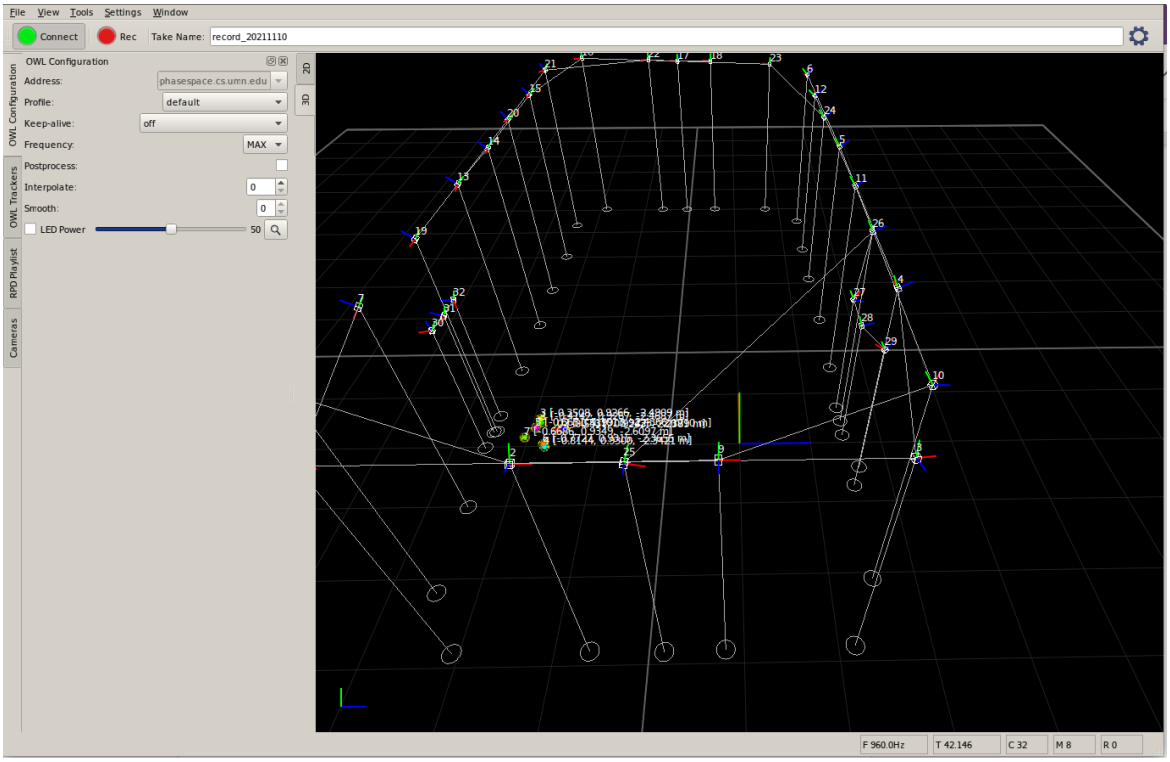
Step 1: Open the Master Client and connect to the Phasespace system

First, install the Master Client. Although it is a Windows-only application, it seems to work well under WINE (a program that makes it possible to run Windows apps on Linux or macOS).

Under the “OWL Configuration” tab, enter cs-phasespace.cs.umn.edu as the address. Then, click connect

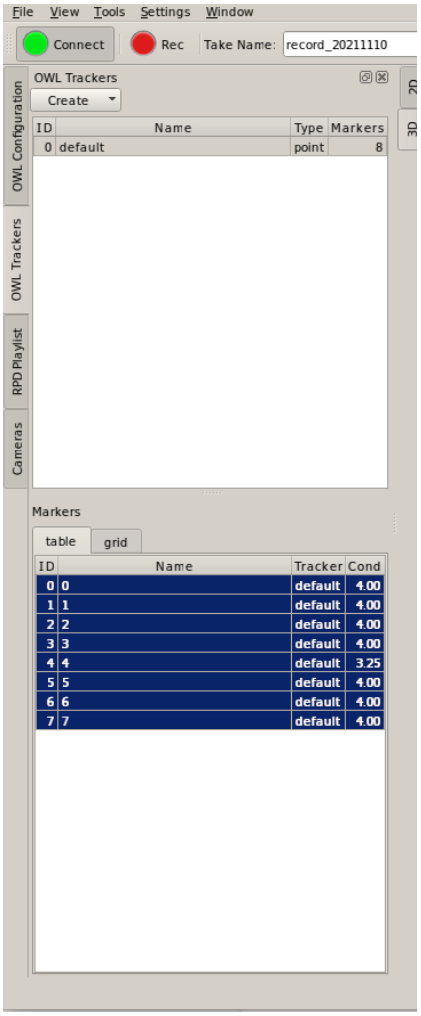


Once you click connect, the button should turn green, and the application should display the locations of all of the cameras, as well as any markers that may be active.

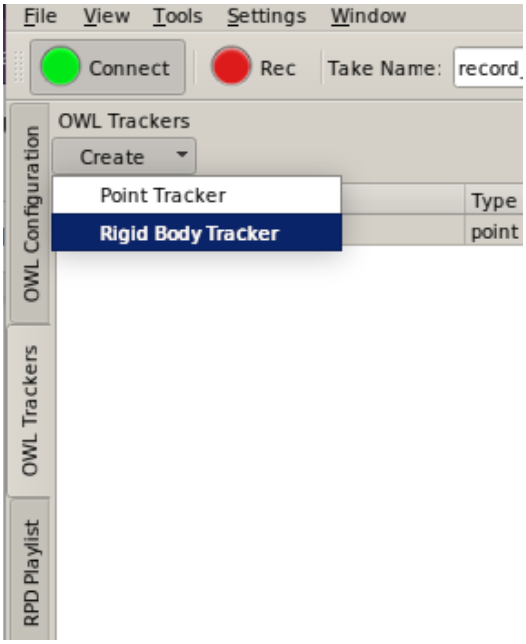


Step 2: Create a rigid body tracker in the Master Client and export the JSON file

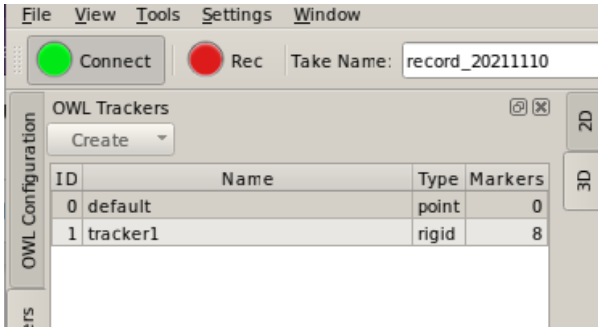
We need to group markers together into a rigid body. This can be done under the “OWL trackers” tab. Select all of the markers that are attached to the rigid body that you want to track. In this example, all 8 markers are firmly attached to our object. Selected markers are highlighted in the GUI



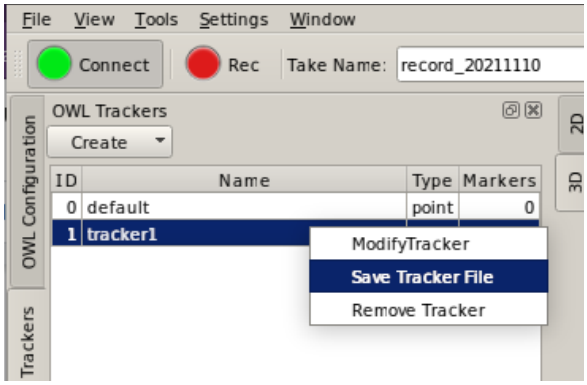
Then, under the “Create” dropdown, click “Rigid Body Tracker”. **When you do this, make sure that your rigid body is aligned to the axes of the world frame!**



This should cause a new entry to appear under the OWL Trackers section



Right click the newly created tracker, and save it to a file



Step 3: Take marker positions from the saved JSON file and put them in your code

Our JSON file looks like this:

```
{
   "trackers": [
```

```
        {
            "id": 1,
            "markers": [
                {
                    "id": 0,
                    "name": "0",
                    "options": "pos=112.331,-23.6597,111.223"
                },
                {
                    "id": 1,
                    "name": "1",
                    "options": "pos=128.5,-26.7466,-150.562"
                },
                {
                    "id": 2,
                    "name": "2",
                    "options": "pos=165.12,-25.5539,162.483"
                },
                {
                    "id": 3,
                    "name": "3",
                    "options": "pos=170.524,-21.5515,-194.869"
                },
                {
                    "id": 4,
                    "name": "4",
                    "options": "pos=-173.485,-21.0011,140.923"
                },
                {
                    "id": 5,
                    "name": "5",
                    "options": "pos=-87.2958,153.926,-8.16226"
                },
                {
                    "id": 6,
                    "name": "6",
                    "options": "pos=-147.692,-20.2545,102.06"
                },
                {
                    "id": 7,
                    "name": "7",
                    "options": "pos=-168.002,-15.1584,-163.098"
                }
            ],
            "name": "tracker1",
            "options": "",
            "type": "rigid"
        }
    ]
}
```


Importantly, it contains the position of each tracker. It is straightforward to take these positions and add code to our ROS node to track our rigid body.

```
    OWL::Context owl;
```

```
if (owl.open(address) <= 0 || owl.initialize("timebase=1,1000000") <= 0)
{
    ROS_ERROR("Could not connect to the address %s", address.c_str());
    return 0;
}
ROS_INFO("successfully connected to %s!", address.c_str());

// connect to phasespace
owl.streaming(1);

// create tracker -- data taken from a JSON file created in the master client
uint32_t tracker_id = 0;
owl.createTracker(tracker_id, "rigid", "drone_rigid_body");

owl.assignMarker(tracker_id, 0, "0", "pos=112.331,-23.6597,111.223");
owl.assignMarker(tracker_id, 1, "1", "pos=128.5,-26.7466,-150.562");
owl.assignMarker(tracker_id, 2, "2", "pos=165.12,-25.5539,162.483");
owl.assignMarker(tracker_id, 3, "3", "pos=170.524,-21.5515,-194.869");
owl.assignMarker(tracker_id, 4, "4", "pos=-173.485,-21.0011,140.923");
owl.assignMarker(tracker_id, 5, "5", "pos=-87.2958,153.926,-8.16226");
owl.assignMarker(tracker_id, 6, "6", "pos=-147.692,-20.2545,102.06");
owl.assignMarker(tracker_id, 7, "7", "pos=-168.002,-15.1584,-163.098");
```

The Phasespace C++ API guide explains what the parameters mean in greater detail. Notice that the marker positions are directly taken from the JSON file above.

Once our code creates the rigid body tracker, the Phasespace system will be able to provide position data (x, y, z coordinates) and orientation data (a quaternion) for our rigid body.

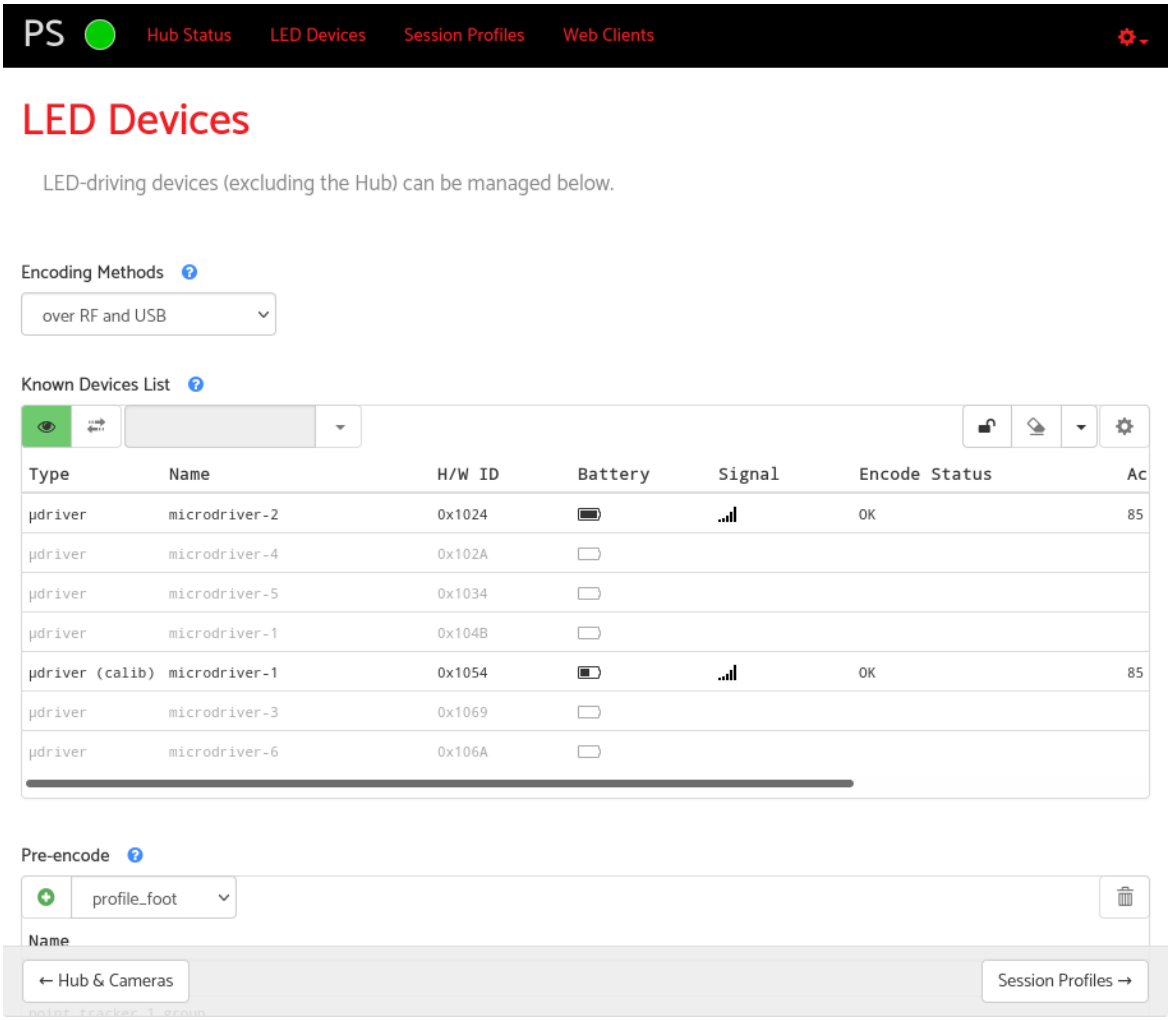
TODO: talk about our phasespace package

Tracking multiple objects with the Phasespace system

It can be useful to track multiple objects using the Phasespace system. As long as you have multiple marker LED strings and microdrivers/dongles, it is relatively straightforward to track multiple objects

Step 1: Ensure Phasespace is aware of your microdrivers

First, ensure that all of the microdrivers that you want to use with the phasespace are visible under the LED Devices table. Microdrivers should be labeled with their H/W ID.



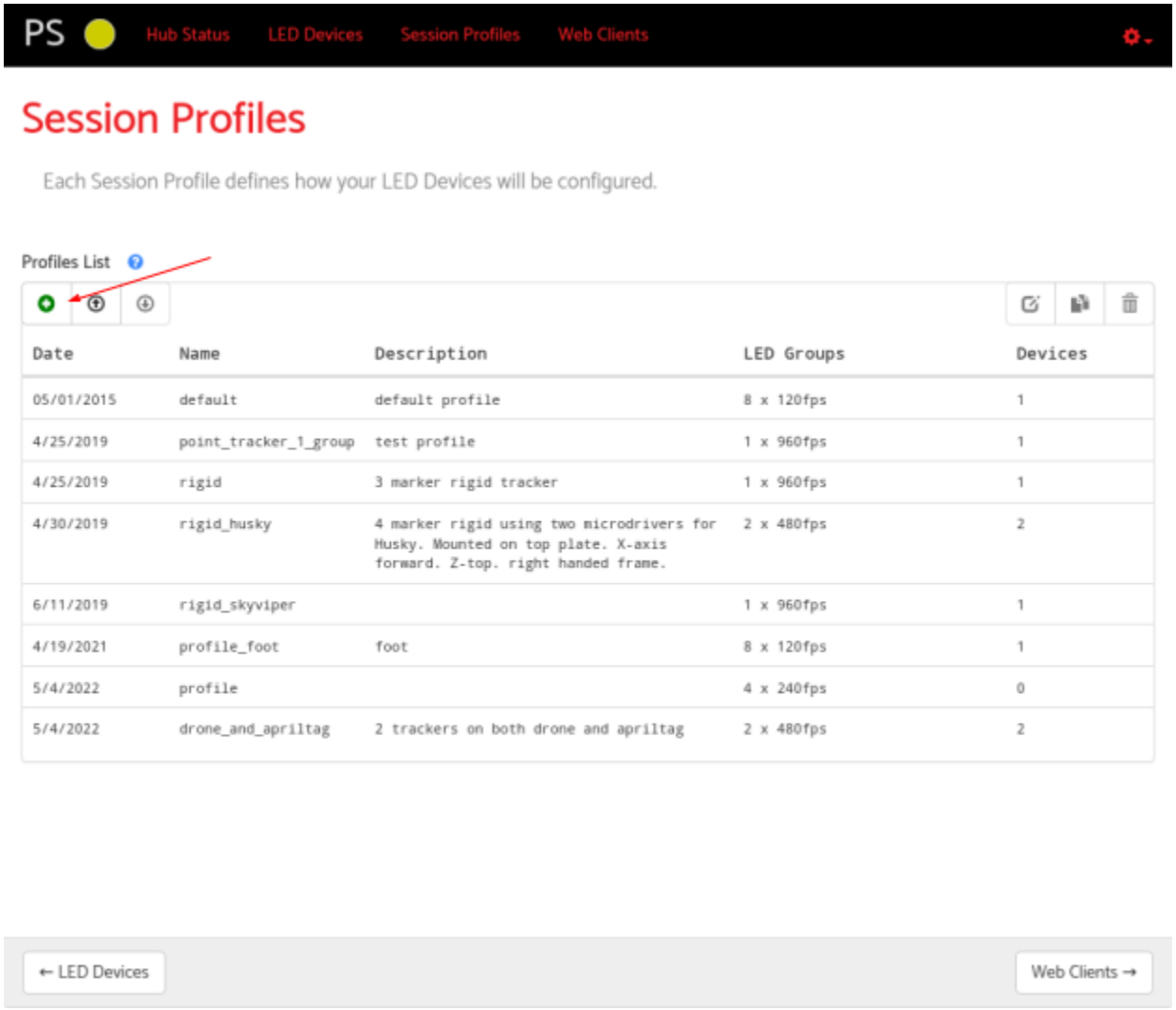
Above: LED devices table

In our case, we would like to track two objects using microdrivers with IDs 1054 and 1024. Both of these drivers are visible in the above table. If the drivers that you would like to use are grayed out or not visible in the table, ensure that they are turned on and that the eye button has been clicked.

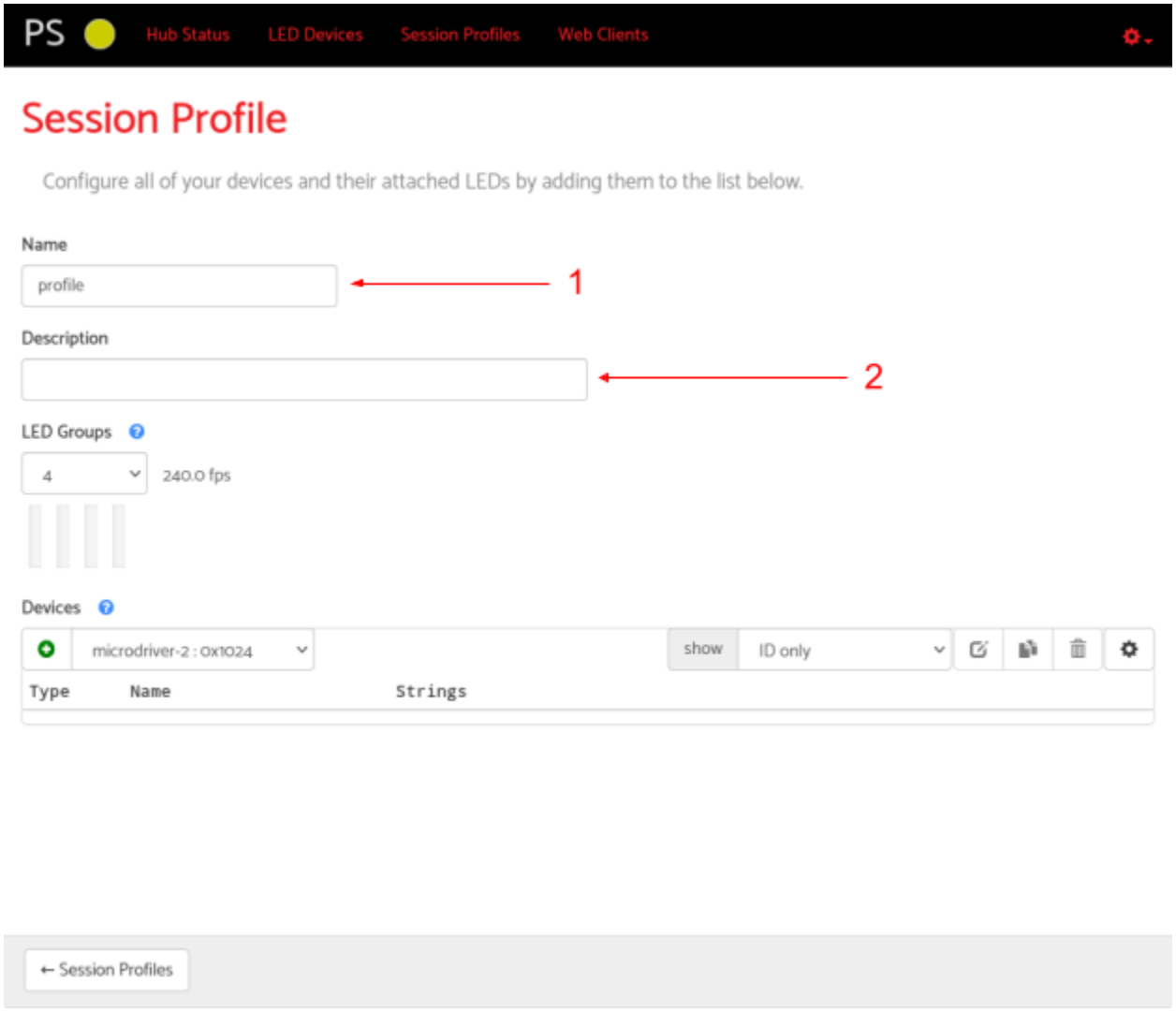
Step 2: Configure Session Profile

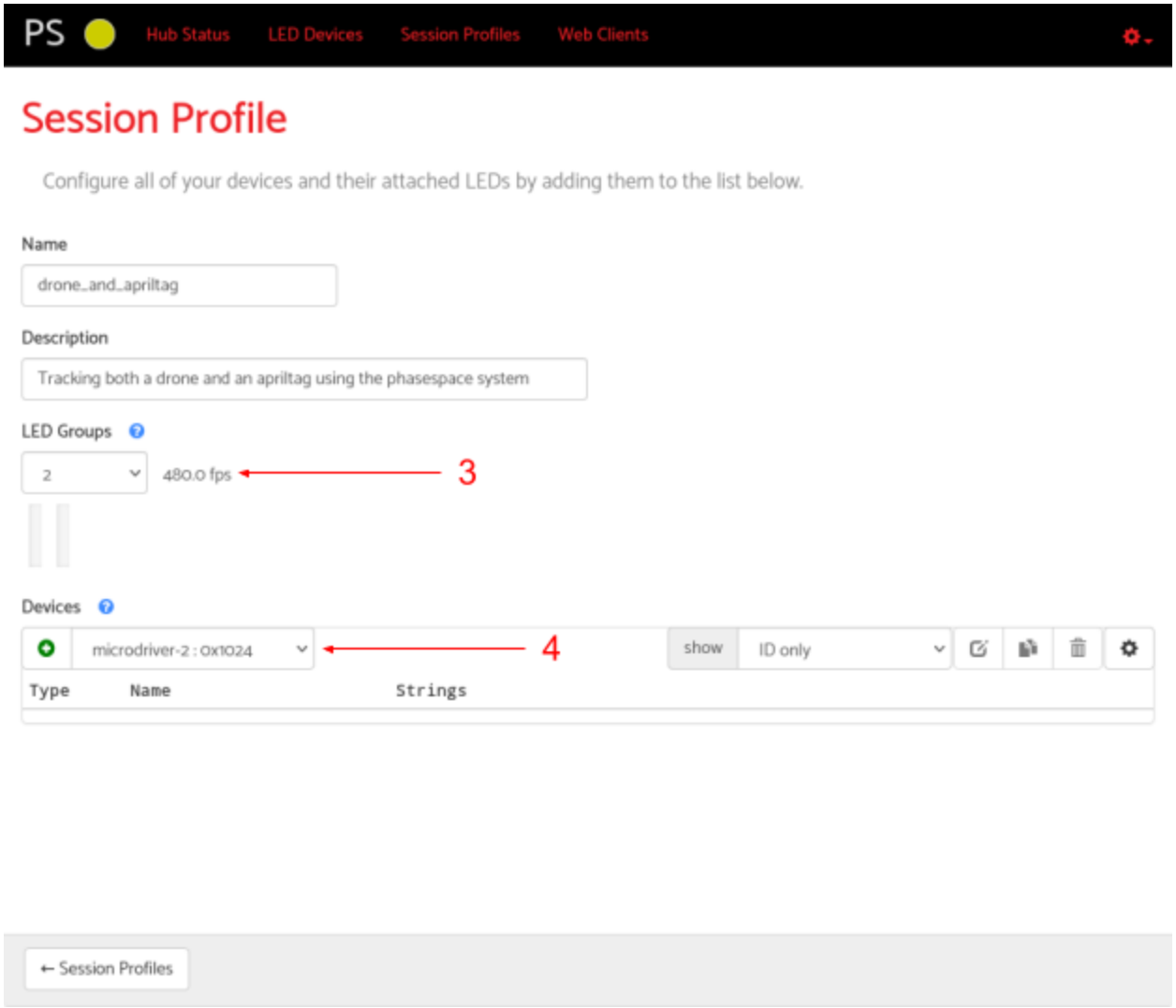
A session profile contains information about what microdrivers you are going to use with the phasespace.

First, navigate to the “Session Profiles” tab in the Phasespace web client. Next, click the plus sign to add a session profile.



Give your session profile a name and description that are distinct from the names and descriptions of other session profiles.

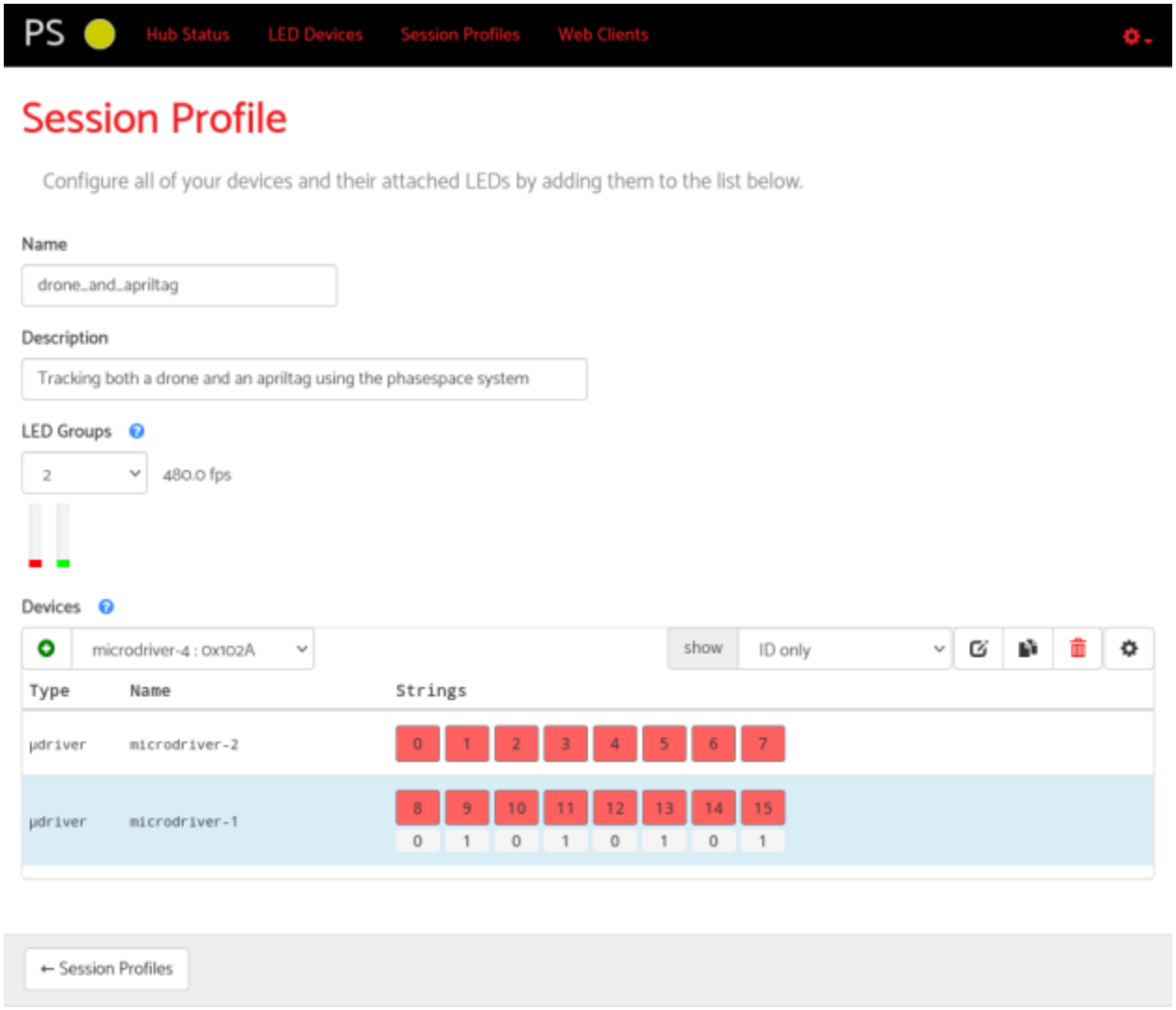




Third, specify how many LED groups your session profile will have using the LED groups dropdown. An LED group is a group of individual LEDs that are turned on together. In order to reduce the likelihood that markers are confused with each other, the Phasespace system actually blinks marker LEDs on and off very fast. There can be a maximum of 8 LEDs in a given group.

If you have LEDs that are close together, then it is a good idea to put them in separate LED groups.

Fourth, use the dropdown to select the microdriver that you would like to use, and then click the plus sign to add it to your session profile.

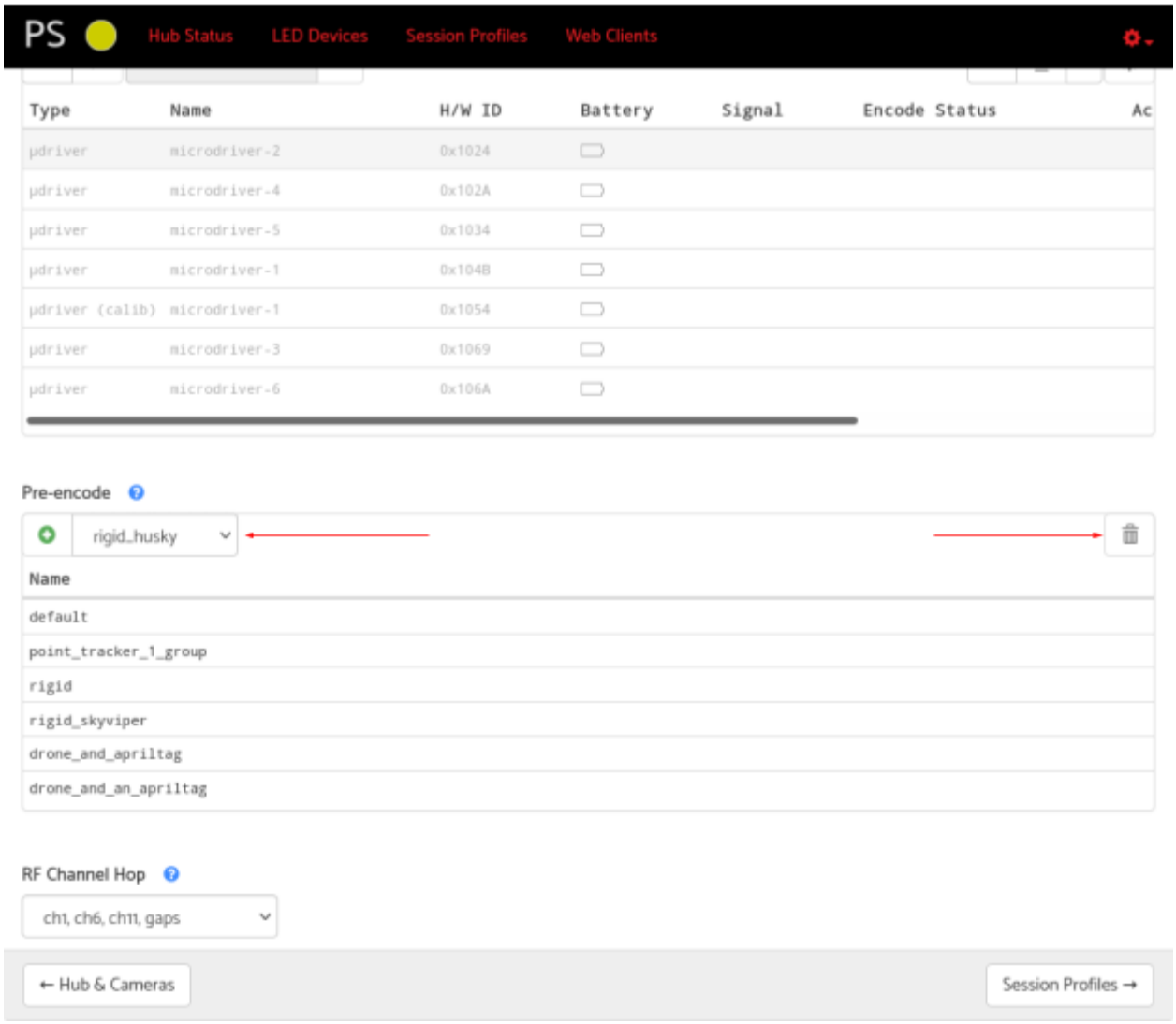


Now that we have added our two microdrivers, our profile looks like this. For our use case, this is sufficient. Depending on your marker arrangement, it may be necessary to twiddle around with how LEDs are grouped together in order to get good tracking results.

If any of your marker IDs (the numbers in the red boxes) are -1, it means that marker will not be used by the phasespace system. If you would like to use it, you need to add more LED groups.

Step 3: Pre-encode Session Profile

Now that we have created our session profile, we need to add it to the list of profiles that we are actively using

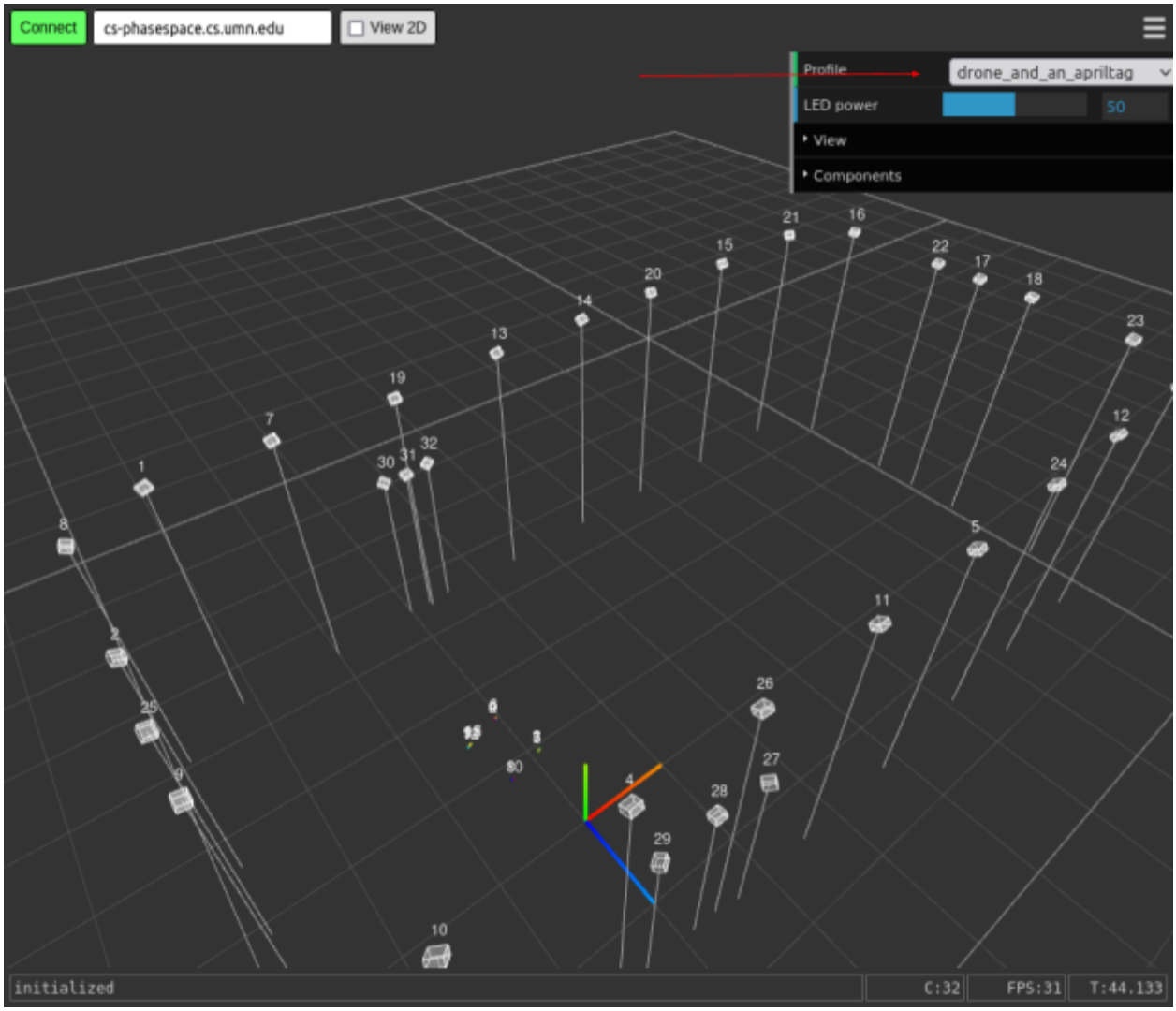


Select your newly created profile in this dropdown, and then click the plus sign to pre-encode it.

Note that it is not possible to pre-encode more than 6 profiles at a time. To remove a profile from this list, select it in the table, and click the trash can button. This is not a destructive action – the profile is not deleted, and the people using that profile can come back and add it to this list at a later time.

Step 4: Use session profile

Now that our profile is pre-encoded, we can enable it in the 3D viewer web client



Click on the profile dropdown and select the profile that you would like to use.

After changing profiles, all LED markers should turn off, followed by the LEDs specified in the profile turning on.