

ALPACA WORKBENCH FOR RASPBERRY PI 11 BULLDOG (32 AND 64 BIT KITS)

Bob Denny (V3.2 April 2023)

Contact on [ASCOM Talk Developer Forum](#)

This document shows you how to build an Alpaca workbench consisting of the ASCOM Alpaca "Omni Simulators" package (**included**), the ConformU conformance checker tool (**included**), and info on the [Python client API library Alpyca](#) and the [Wireshark](#) net analyzer tool to analyze the HTTP/REST network data. Since this is (primarily) a developer package, help will be provided on the [ASCOM Driver and Application Development Forum](#).

Please note that some of the screen shots are for older versions of ConformU and will be updated in a future version of this document.

CONTENTS

| | |
|--|----|
| INSTALLATION | 2 |
| UPDATING THE WORKBENCH | 2 |
| USING UPDATE AVAILABLE LINKS IN THE SIMULATORS AND CONFORMU | 2 |
| INITIAL START AND CONFIGURATION OF SIMULATORS | 3 |
| Setting up the Host App (Server) | 4 |
| Setting up Devices | 6 |
| SIMULATORS ROUTINE USE - ALPACA | 7 |
| SIMULATORS ROUTINE USE - FROM WINDOWS APPS VIA A DYNAMIC DRIVER | 10 |
| SIMULATORS USAGE FROM WINDOWS APPS THAT DON'T USE THE CHOOSER (SGP, NINA, ETC) | 12 |
| Chooser in ASCOM Diagnostics | 12 |
| ADVANCED ALPACA USAGE FROM WINDOWS APPS..... | 13 |
| EXPLORING ALPACA FROM PYTHON 3 - THE ALPYCA PACKAGE..... | 14 |
| EXPLORING ALPACA AND THE API WITH THE OPENAPI BROWSER | 15 |
| USING CONFORMU - THE UNIVERSAL CONFORMANCE CHECKER | 17 |
| Using the Alpaca Protocol Checker..... | 20 |

| | |
|---|----|
| Using the Discovery Mapping Tool | 21 |
| Discovery Diagnostics | 22 |
| EXAMINING HTTP/REST MESSAGES WITH WIRESHARK | 22 |
| Installing Wireshark and Setting Privileges..... | 23 |
| Checking the Wireshark Installation..... | 23 |
| Setting up a Test & Learning Environment | 24 |

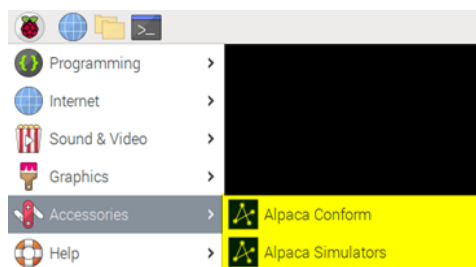
INSTALLATION

The Omni Simulators and ConformU components are placed into `/opt1/ascom.alpaca.simulators[64]` and `/opt/ascom.alpaca.conformu[64]`, respectively, along with the .desktop files needed in `/usr/share/applications` for the simulators app to appear in the Desktop GUI's Berry menu under Accessories.

The archive contains kits for both the 32-bit and 64-bit Bulldog versions. The 32-bit build will work on the older Buster OS as well but will not work on 64 bit Bulldog. These two builds are in the **Bulldog32** and **Bulldog64** folders, respectively. Open a bash shell and change directory to the BulldogXX folder that's appropriate to your OS then run the **install.sh** or **install64.sh** shell script, for example.

```
./install.sh
```

After it completes the Berry menu should show **Alpaca Conform** and **Alpaca Simulators** with '(64)' on a 64 bit system:



To uninstall use the included `uninstall.sh` scripts.

UPDATING THE WORKBENCH

If you want to install a newer version of the Alpaca Workbench, follow the installation instructions above, except first run the `uninstall.sh` script to remove the old files, and then `install.sh` to install the new ones. This will avoid leaving fossil files around.

USING UPDATE AVAILABLE LINKS IN THE SIMULATORS AND CONFORMU

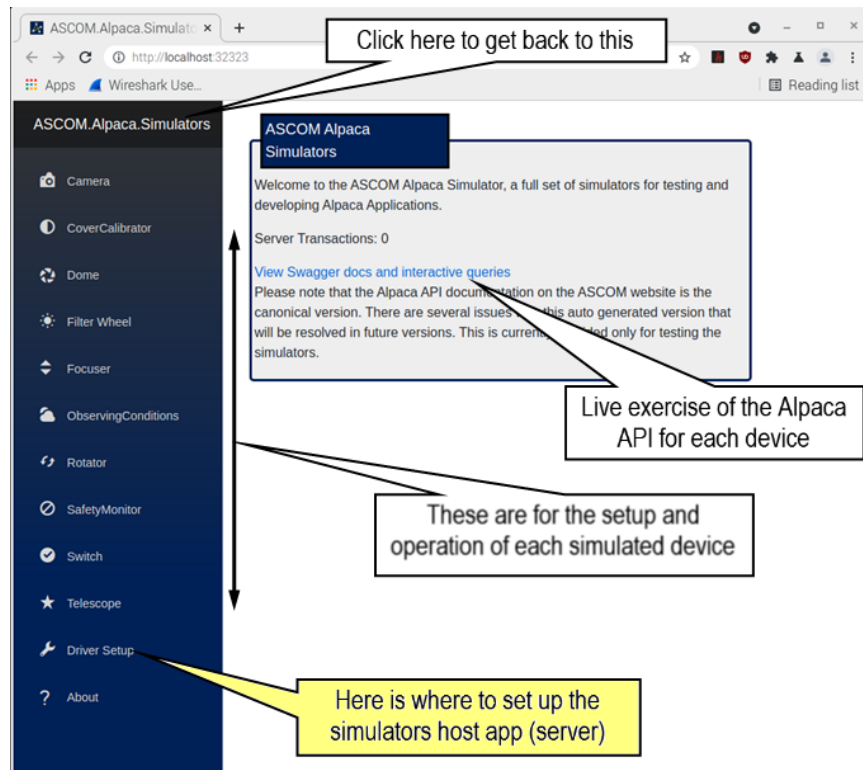
¹ Considered best practices, see [What does /opt mean in Linux](#)

If you see an Update Available link in the Simulators or in ConformU it will take you to the GitHub "latest release". The updating process is not automatic because these apps are compatible with many Linux distros and are installed by non-Workbench users. It is up to you to extract the new version's files to the locations shown above. Be aware that they are meant to be installed with root:root so you'll need root/sudo privs to do this.

INITIAL START AND CONFIGURATION OF SIMULATORS

Selecting the Alpaca Simulators menu item should cause the Omni Simulator to start. You will first see a shell window containing several logged messages and, depending on your browser, possibly several harmless GUI and other messages. Shortly thereafter your default browser (normally Chromium) will appear.

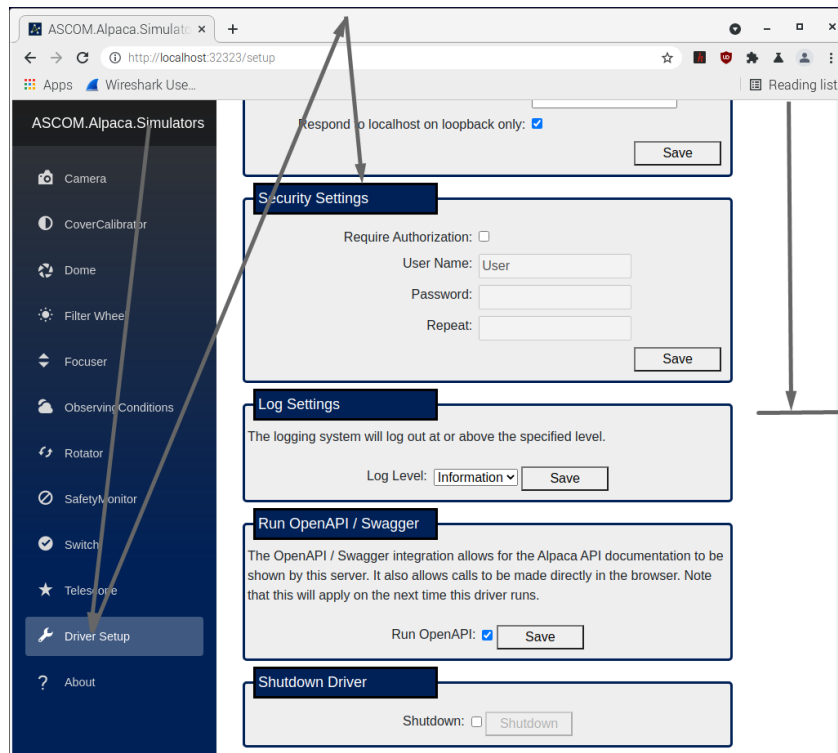
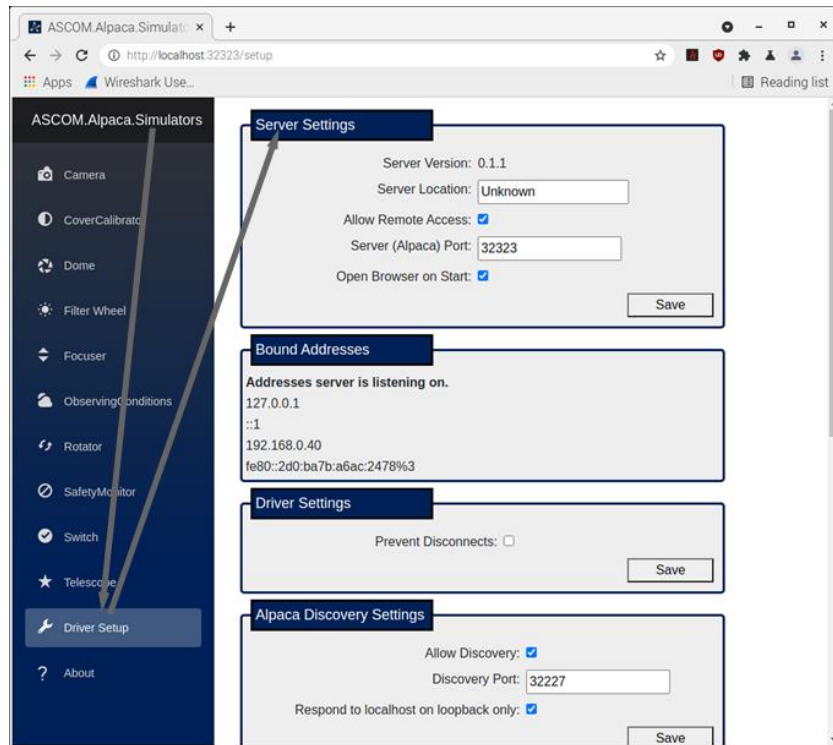
```
pi@rpil:/opt/ascom.alpaca.simulators
File Edit Tabs Help
pi@rpil:/opt/ascom.alpaca.simulators $ ./ascom.alpaca.simulators-aarch64.AppImage
ASCOM Alpaca Simulators version 0.3.0-preview01
11/9/2022 5:24:38 PM [Information] - ASCOM Alpaca Simulators version 0.3.0-preview01
No startup url args detected, binding to saved server settings.
11/9/2022 5:24:38 PM [Information] - No startup url args detected, binding to saved server settings.
Startup URL args: --urls=http://*:32323
11/9/2022 5:24:38 PM [Information] - Startup URL args: --urls=http://*:32323
Starting Discovery on port: 32227
info: Microsoft.Hosting.Lifetime[14]
Now listening on: http://[::]:32323
11/09/2022 17:24:41 [Information] - ASCOM Alpaca Simulators Starting Services
info: Microsoft.Hosting.Lifetime[0]
Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
Content root path: /tmp/.mount_ascom.k25F15/usr/bin
11/09/2022 17:24:46 [Information] - Camera 0 - UUID of 5e2d287a-64bf-4e19-8d91-f2795048a8fc
11/09/2022 17:24:46 [Information] - CoverCalibrator 0 - Starting initialisation
11/09/2022 17:24:46 [Information] - CoverCalibrator 0 - UUID of 4e60dcfc-4c4c-4cf0-8b25-106b6b9df2a0
11/09/2022 17:24:46 [Information] - CoverCalibrator 0 - Set calibrator timer to: 2000ms.
11/09/2022 17:24:46 [Information] - CoverCalibrator 0 - Set cover timer to: 5000ms.
11/09/2022 17:24:46 [Information] - CoverCalibrator 0 - Completed initialisation
11/09/2022 17:24:46 [Information] - Dome 0 - UUID of 382dd95c-518d-460c-a696-49df27585b8b
11/09/2022 17:24:46 [Information] - FilterWheel 0 - Starting initialization
11/09/2022 17:24:46 [Information] - FilterWheel 0 - UUID of bed2779f-0564-465c-b774-ce90af3e0a71
11/09/2022 17:24:46 [Information] - Focuser 0 - UUID of 420e61fc-9e4b-41a2-bce3-dfe5a4202c72
11/09/2022 17:24:46 [Information] - ObservingConditions 0 - UUID of cb2df31d-96e0-4c61-93b5-2dd355783d3b
11/09/2022 17:24:46 [Information] - Rotator 0 - UUID of ee7cb460-1f14-4b25-83bd-7393588cd1f0
```



Note the URL of the simulator's home page `http://localhost:32323`. You might want to bookmark this.

SETTING UP THE HOST APP (SERVER)

The first thing to do is open and review the Driver Setup window. All the simulators share a single host app (server) and the common settings for this are here.

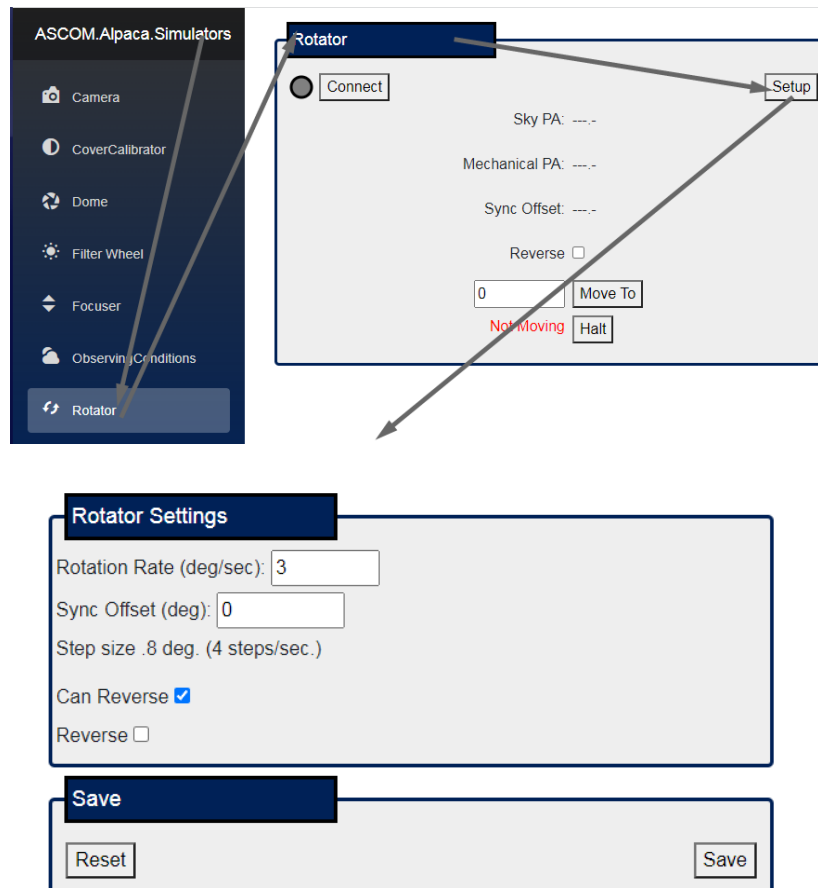


For now, just fill in the Server Location with any descriptive text you wish and click Save. Allow Remote Access controls whether other computers on your LAN will have access to the simulated devices. Leave the Server (Alpaca) Port as well as the Discovery Port at their default values unless you have a good reason (and the network

configuration skills) to change them. The other settings are for advanced/unusual situations and are self-explanatory.

SETTING UP DEVICES

Each simulated device has a SetupDialog() page which you can display with its Setup button. This serves the same role as the SetupDialog() window that an ASCOM/COM driver has. The device characteristics are established on this page. The settings you make are persistent and apply to all apps that use the simulated device. For example, to set up your simulated rotator, opening its SetupDialog() window:

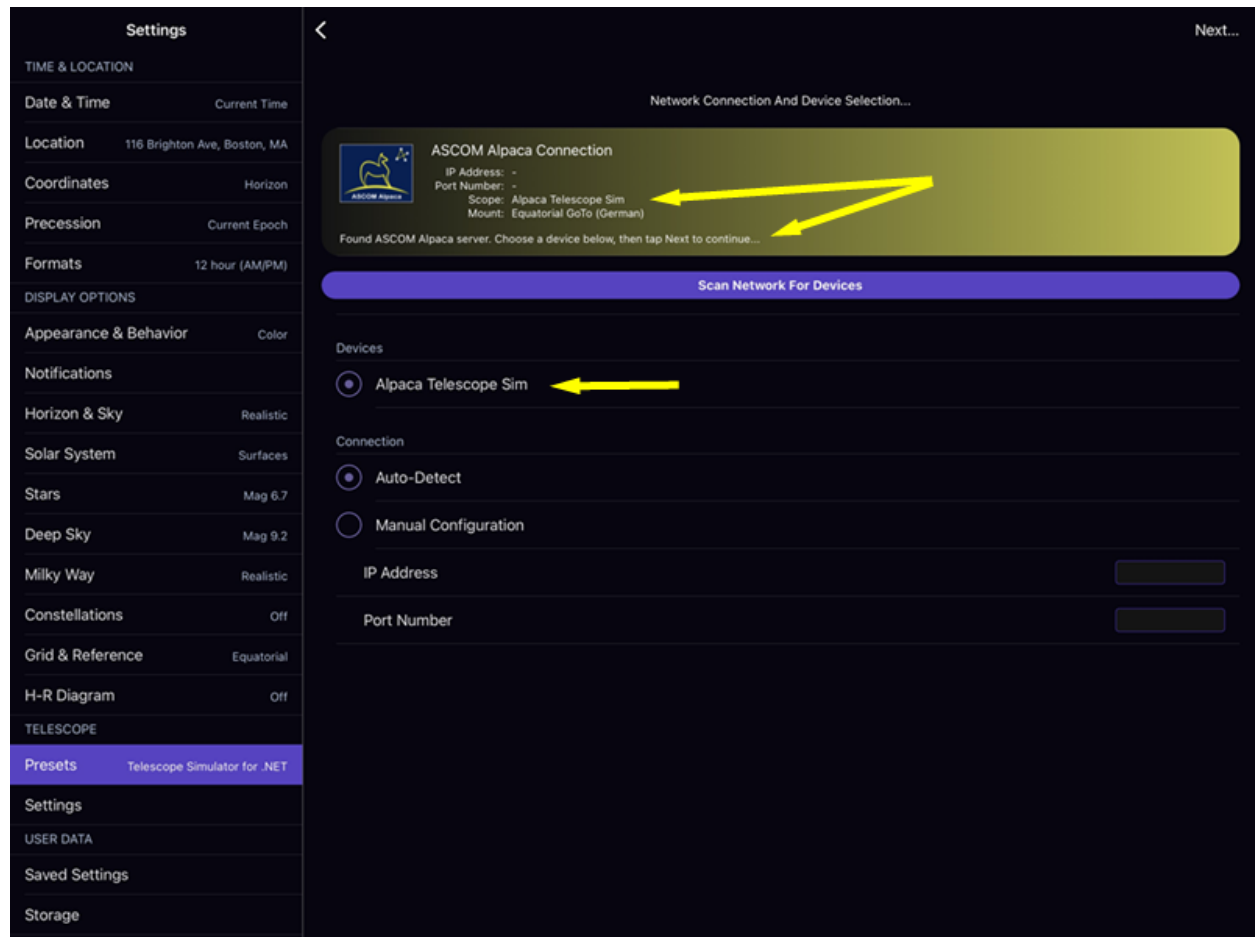


Clicking Save makes your changes persistent. Use the browser's back arrow to close the setup dialog. Hopefully this will get you going with the simulators you want to use.

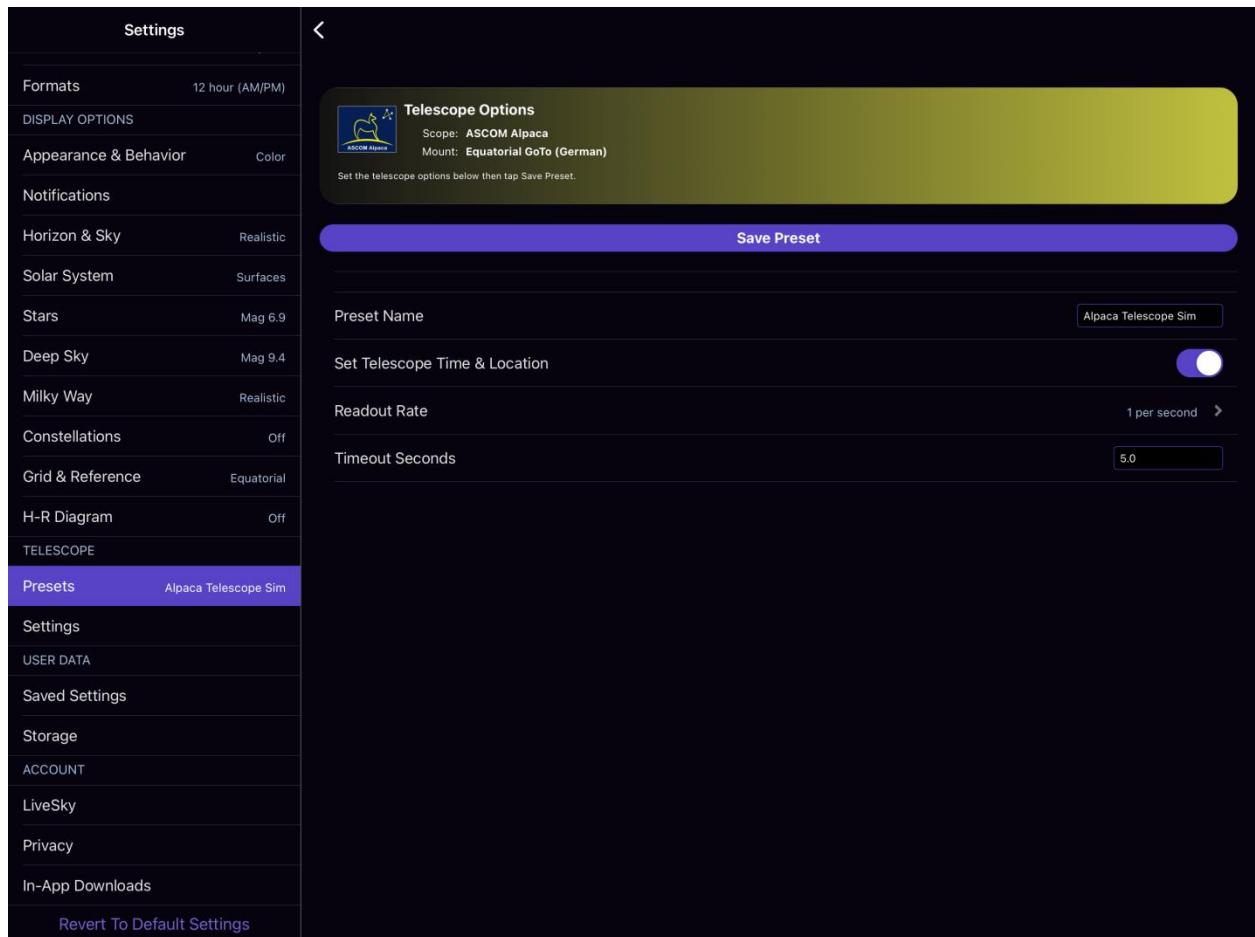
SIMULATORS ROUTINE USE - ALPACA

Please note: Windows is not required for Alpaca as we will see.

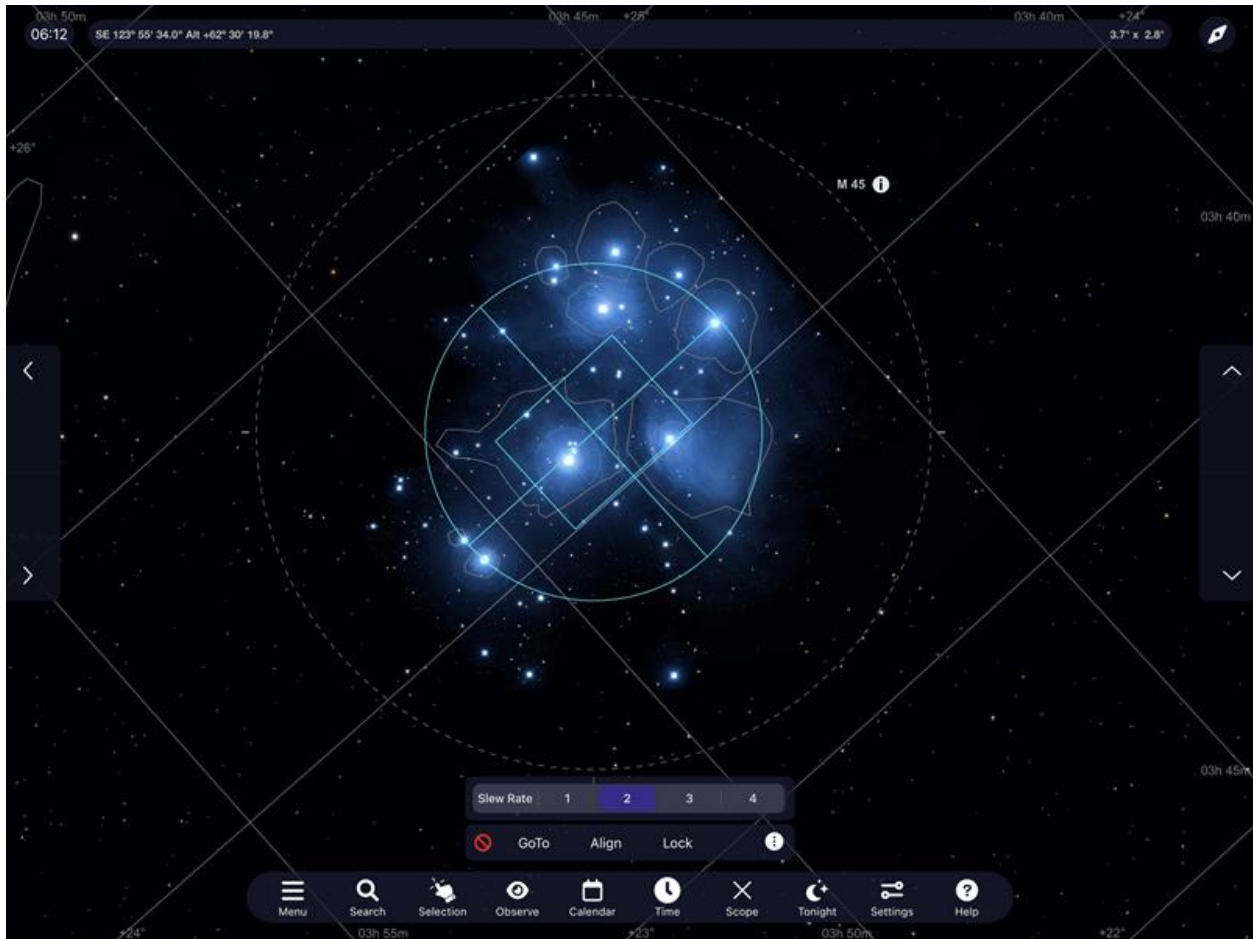
Once the simulators are running, you can close the browser and the simulators should simply answer and provide the simulated devices to any app that wants to use them. For example, using [SkySafari 7 Plus or Pro](#), make a new Telescope Preset, select ASCOM Alpaca as the Connection Type. Leave Auto-Detect and click Scan. You may need to scan twice. You should end up with this:



Then Next... Leave Set Time and Location on. SkySafari will send the location (latitude and longitude) to the (simulated) telescope, and it will be a persistent simulator setting. The simulator's date/time will not be changed because the simulator does not have root privileges.



At this point you'll have a simulated mount in SkySafari. Connect in SkySafari, and the sky should move to center on the simulator telescope coordinates. You can also open the Telescope settings in the browser and look to see that the Latitude and Longitude of the simulator (now) matches the location you have set in Sky Safari. This will only be true after you have connected SkySafari to the telescope. That's when it sends the coordinates. Here's SkySafari connected to the simulated telescope pointing at the Pleiades and showing the configured field of view indicator.

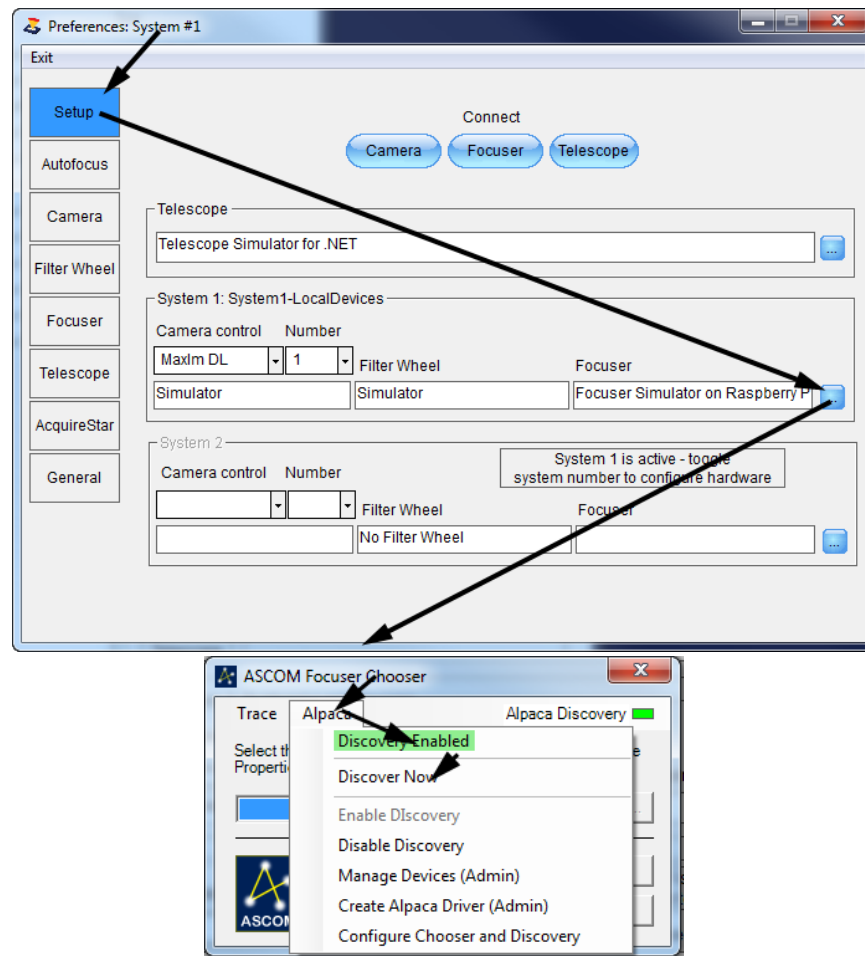


SIMULATORS ROUTINE USE - FROM WINDOWS APPS VIA A DYNAMIC DRIVER

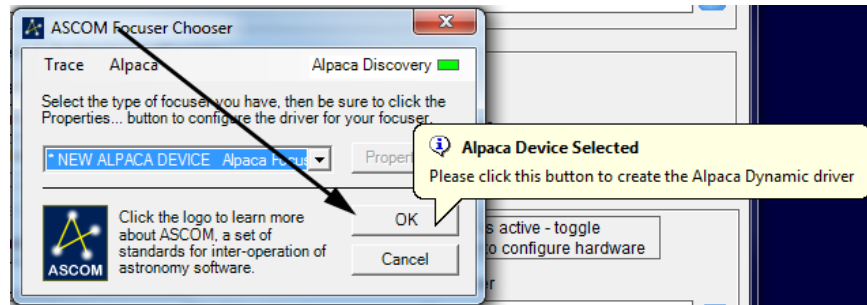
Please note: Windows is not required for Alpaca as we saw above.

The Windows ASCOM Platform provides a Chooser that can automatically create and configure an ASCOM driver that translates from COM to Alpaca and back. This means your Windows astronomy app can use Alpaca devices (like the OmniSimulator devices) today with **no changes to the app**.

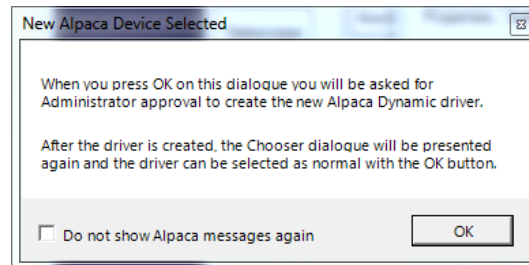
You only need to do this once for this Alpaca device. In your ASCOM-compatible astronomy app, use the Chooser as usual to specify the type of device. Let's say you are using FocusMax, and your focuser is a self-contained WiFi Alpaca device. You can use the Omni Simulator's WiFi Alpaca focuser to simulate this. Assuming your Windows system is on the same local network, in FocusMax show the ASCOM Focuser Chooser:



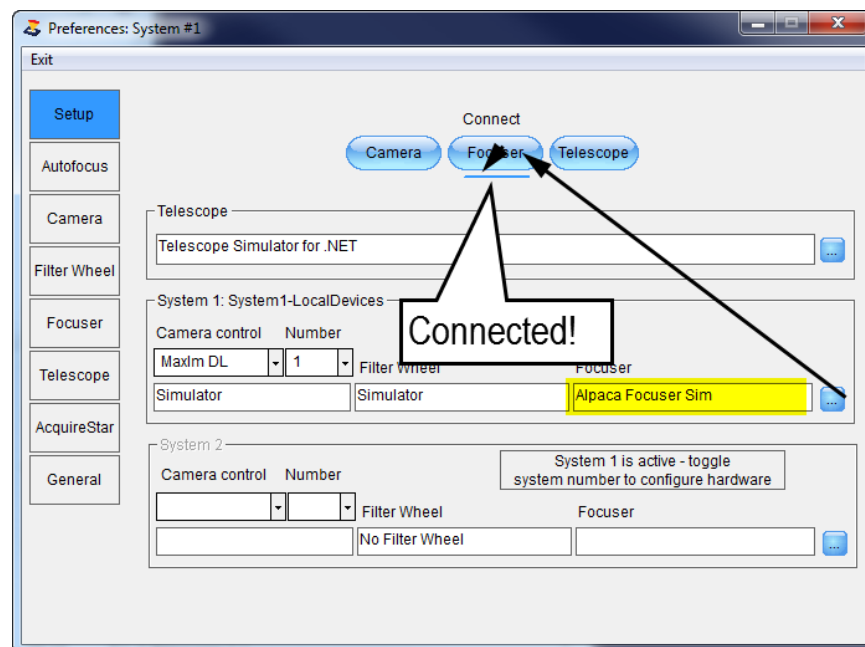
Enable Discovery (one time setting) then Discover Now. The Chooser will automatically discover when opened the next time and thereafter. You will see the Alpaca Discovery light flash on and off. Once it is finished, click the device list as usual, you will see a new Alpaca-discovered Focuser listed (it's the simulator). Select it. You'll see this tooltip. Click OK as advised.



Now you'll be asked for Administrator privileges, needed to create a new ASCOM/COM device that is accessible to all users not just you. Allow it, and probably tell the Platform not to bug you about this in the future:



The new ASCOM COM driver that provides access from your app (in this case FocusMax) to the WiFi Alpaca focuser has been created. **At this point you can click OK and then OK in the Chooser and "just use it" in FocusMax.**



SIMULATORS USAGE FROM WINDOWS APPS THAT DON'T USE THE CHOOSER (SGP, NINA, ETC)

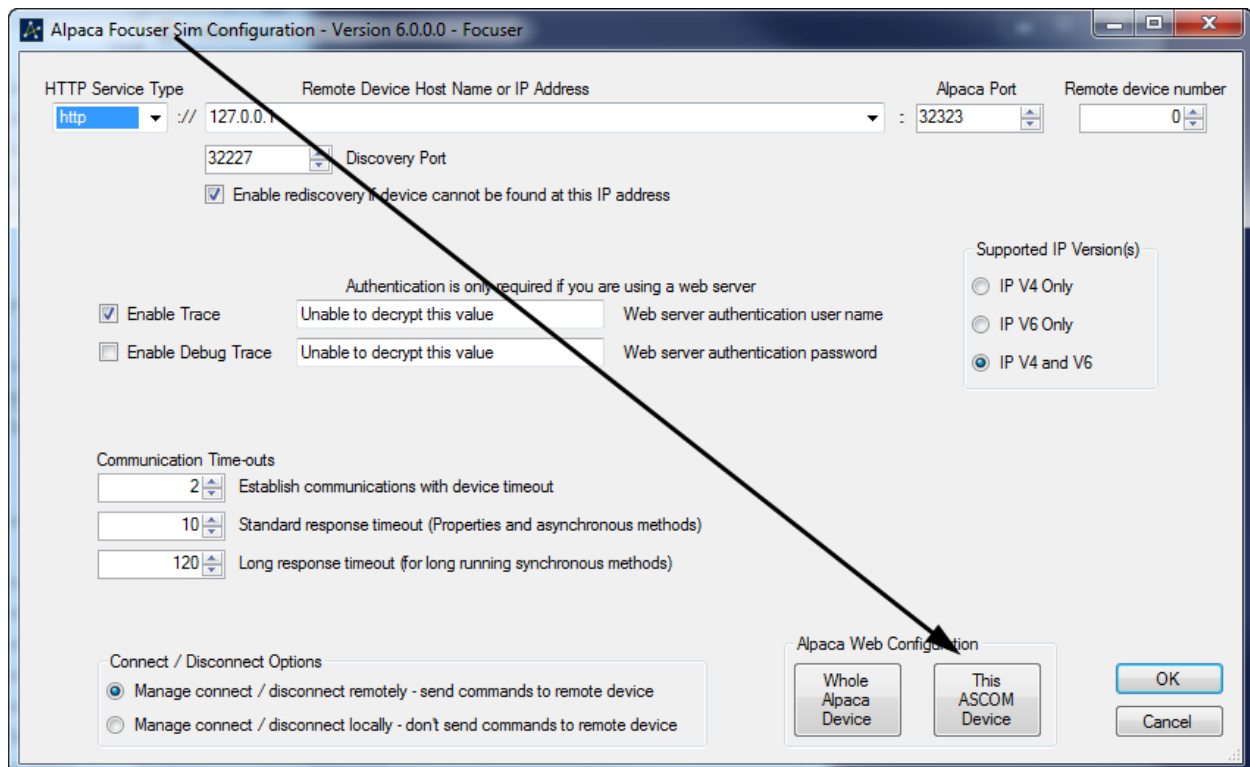
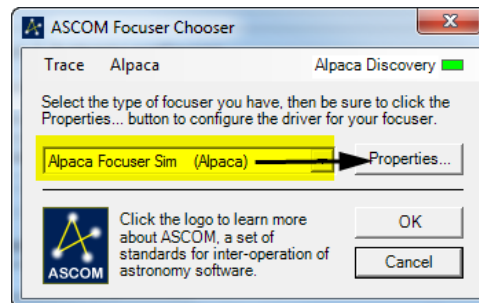
Programs such as Sequence Generator Pro (SGP) and Nighttime Imaging 'N' Astronomy (NINA) don't use the ASCOM Chooser to select their ASCOM devices. Therefore they do not have in-app access to the automatic Alpaca gateway driver creation feature in the ASCOM Chooser. This is no problem though. You can use any program that has Chooser Support *for the type of device to which you want to connect*. In this example the program would need to have Chooser support for a Focuser. We just created the dynamic driver for the Alpaca focuser in FocusMax and it would already show in SGP and NINA. That is enough, but...

CHOOSER IN ASCOM DIAGNOSTICS

If the device you want to use from SGP or NINA is used **only** in SGP or NINA, you can still get a Chooser for the device. Let's say your program needs to use an Alpaca Dome and no other program on your Windows system uses Domes. Go to the Windows Start Menu > ASCOM Platform 6 > Tools > ASCOM Diagnostics. In the ASCOM Diagnostics window that appears, select the Choose Device menu, Choose and Connect to Device (32-bit application). Now you see the Device Connection Tester, but you're not going to connect, you just need the Chooser. Select your device type, then click Choose. The ASCOM Chooser for that device type will appear. Use it to discover and select an Alpaca device, click OK then just close ASCOM Diagnostics. At this point the Alpaca device should appear in the SGP or NINA device list since it is "just another Windows device" as far as those programs are concerned.

ADVANCED ALPACA USAGE FROM WINDOWS APPS

Any time you open the Chooser and one of the Alpaca drivers is selected, you can click **Properties...** to show a window where you can control advanced network capabilities for your app, including username/password security, ports, etc.



If you are network-skilled most of this will make sense. If not don't worry about it. But there is one cool thing you can do from here. If you click "This ASCOM Device", you'll see the device (Focuser in this case) setup page in the browser. Try it. Once you see this and understand what this does, click OK. Similarly, the "Whole Alpaca Device" button will open the browser to show the Driver (the host app and server) setup you saw during initial configuration. Try this now as well.

EXPLORING ALPACA FROM PYTHON 3 - THE ALPYCA PACKAGE

The ASCOM Initiative has released a Python 3 client interface package called Alpyca. You can explore this package via the online docs at [Alpyca: API Library for Alpaca](#) (also [available as a PDF](#)). To install into your Python environment (or virtual environment), as usual:

```
pi@raspberrypi:~ pip install alpyca
```

You can write simple Python scripts to exercise the various simulated devices provided by the Omni Simulator:

```
import time
from alpaca.telescope import *      # Multiple Classes including Enumerations
from alpaca.exceptions import *     # Or just the exceptions you want to catch

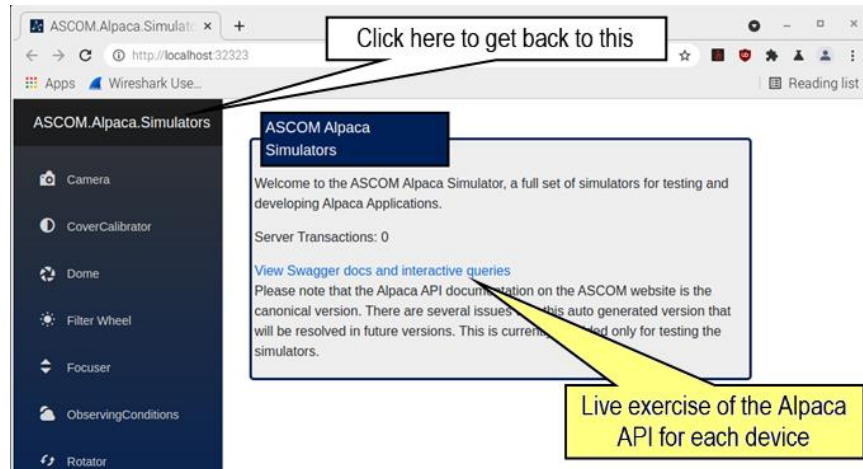
T = Telescope('localhost:32323', 0) # Local Omni Simulator
try:
    T.Connected = True
    print(f'Connected to {T.Name}')
    print(T.Description)
    T.Tracking = True                # Needed for slewing (see below)
    print('Starting slew...')
    T.SlewToCoordinatesAsync(T.SiderealTime + 2, 50)    # 2 hrs east of meridian
    while(T.Slewing):
        time.sleep(5)                # What do a few seconds matter?
        print('... slew completed successfully.')
        print(f'RA={T.RightAscension} DE={T.Declination}')
        print('Turning off tracking then attempting to slew...')
        T.Tracking = False
        T.SlewToCoordinatesAsync(T.SiderealTime + 2, 55)    # 5 deg slew N
        # This will fail for tracking being off
        print("... you won't get here!")
except Exception as e:               # Should catch specific InvalidOperationException
    print(f'Slew failed: {str(e)}')
finally:                             # Assure that you disconnect
    print("Disconnecting...")
    T.Connected = False
```

Results:

```
Connected to Alpaca Telescope Sim
Software Telescope Simulator for ASCOM
Starting slew...
... slew completed successfully.
RA=10.939969572854931 DE=50
Turning off tracking then attempting to slew...
Slew failed: SlewToCoordinatesAsync is not allowed when tracking is False
Disconnecting...
done
```

EXPLORING ALPACA AND THE API WITH THE OPENAPI BROWSER

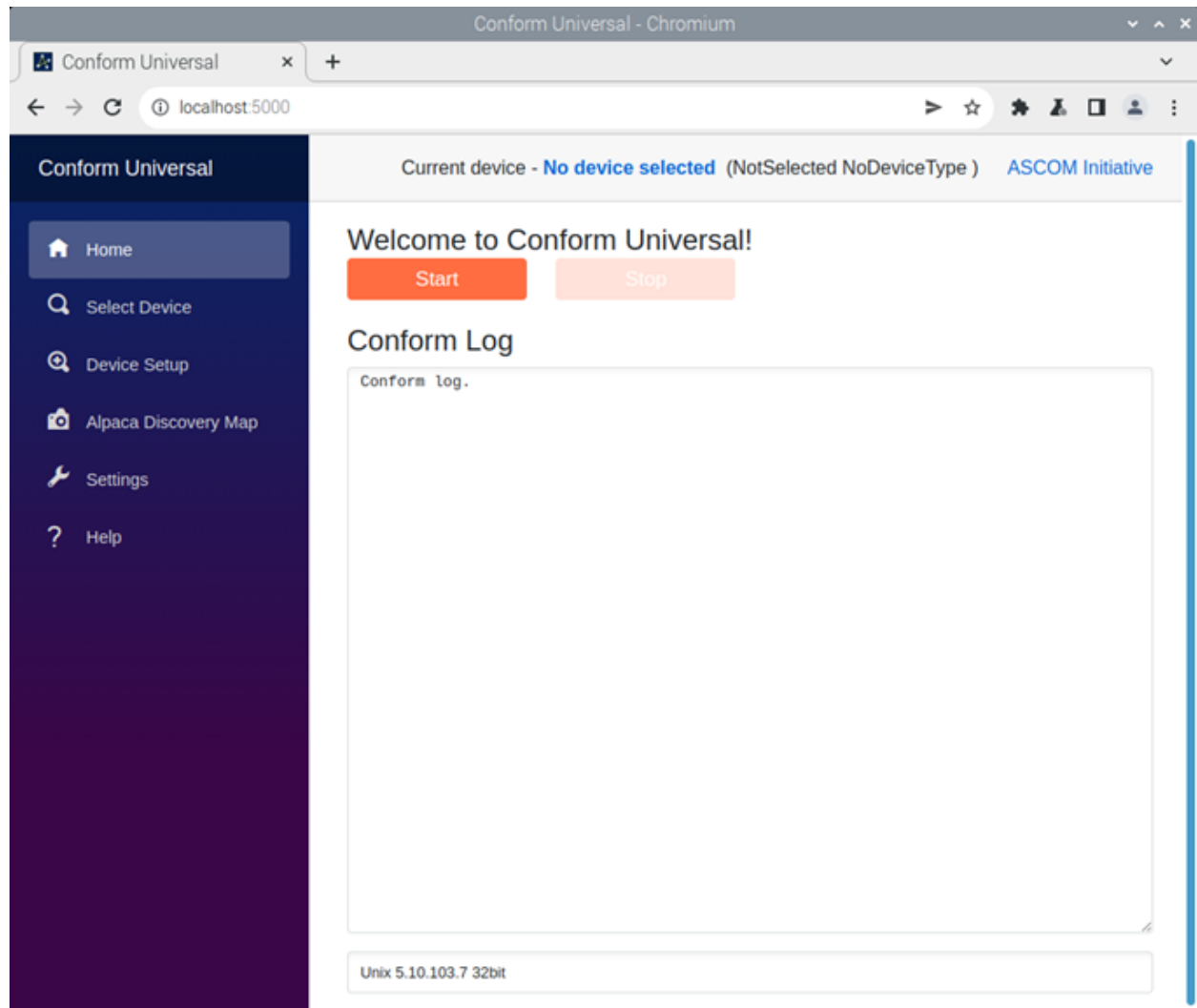
The Omni Simulator (ASCOM.Alpaca.Simulators) includes a complete implementation of the [OpenAPI Browser](#) (formerly known as Swagger UI). All the Alpaca device APIs are presented as visual documentation, including live "Try it out" capability. To get there, return to the simulator's home page.



Eventually, you'll see this. Note that this can be quite slow to initially open on a Raspberry Pi 3. Once it is open each end point should open in less than 10 seconds. On a Raspberry Pi 4 this is much more tolerable. It's beyond the scope of this document to describe how to use this popular API tool. [This Perforce blog post](#) does a decent job of guiding you through the OpenAPI/Swagger UI testing tool.

USING CONFORMU - THE UNIVERSAL CONFORMANCE CHECKER

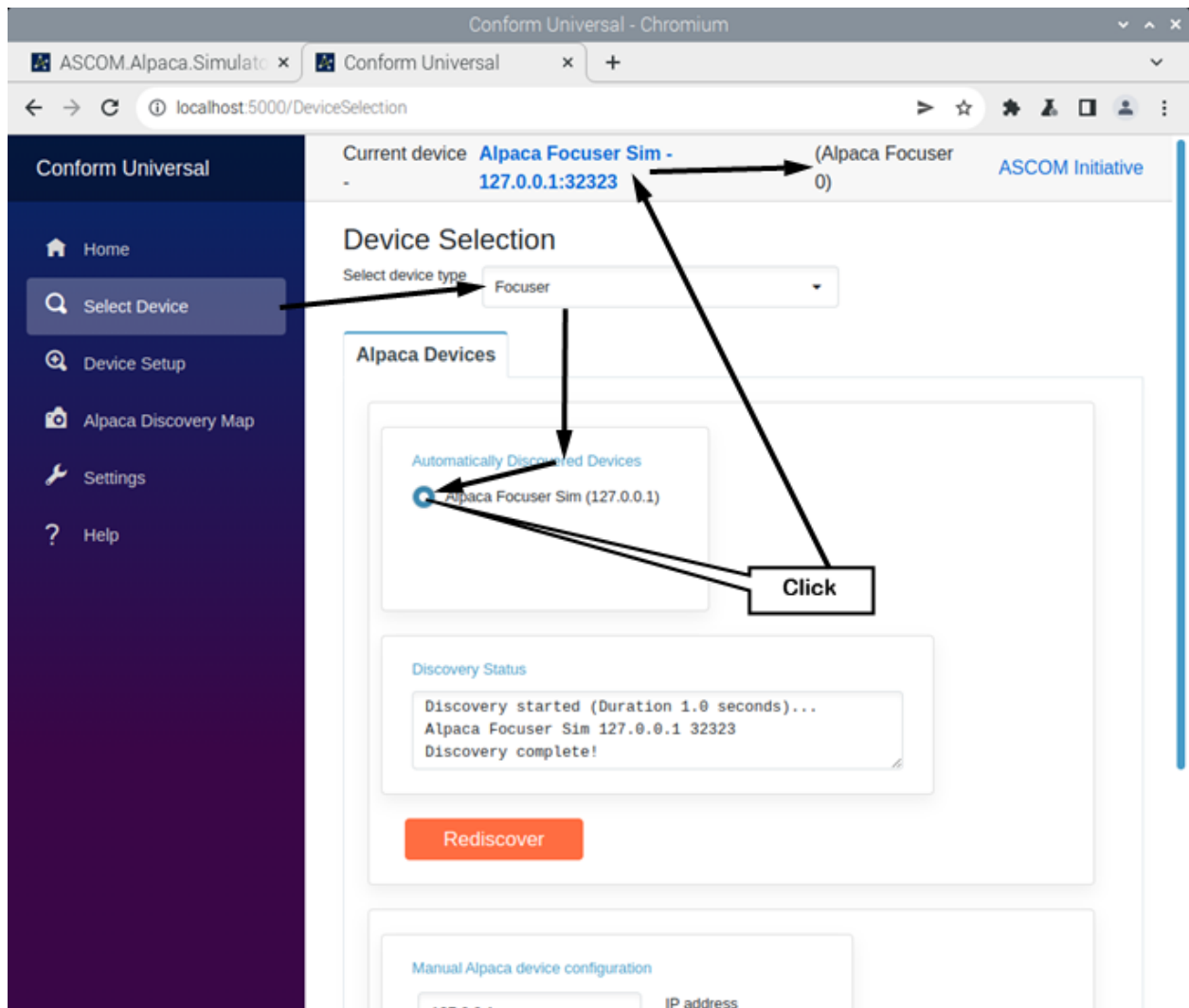
ConformU is a universal cross-platform interface and behavior checker for all Alpaca devices. ConformU performs deep checks of both interface compliance and behavior by interacting with a device through Alpaca², as well as the lower-level Alpaca protocol at the HTTP/JSON level. It also has a useful Alpaca Discovery Map function which allows you to browse Alpaca devices on your local network³. It is quite simple to use, however there are advanced settings available for unusual situations. We'll skip these for now. When you start ConformU via the Berry menu, you should see a shell window similar to the one for the OmniSim, and then this:



If you want to follow along with this, start the Workbench Alpaca Simulators now (you'll see its tab in the Chromium screenshots below). To run a test, you must first tell ConformU which Alpaca device you want to test. Start by clicking the Select Device button and following the map shown below:

² ConformU gets its name "universal" because it can check both ASCOM COM and ASCOM Alpaca, and it also runs on Linux, Windows, and MacOS. However here on the Raspberry Pi, it is limited to checking Alpaca devices.

³ Alpaca discovery is limited to the local network by design.



You're ready to test now. You can get to the device's settings as well as its Alpaca server's settings with the Device Setup link on the left, but we'll assume you have already configured your device to be ready to use. It must be powered up, COM ports correct, etc. **Please note that ConformU will move / operate all devices e.g. slew a telescope to various positions, rotate a dome and open/close its shutter, etc., so please ensure the area is clear of obstructions, including people, and that it is safe!** Now switch back to Home (if needed, a device change will do it for you) and start the test by clicking the Start button:

Conform Universal

Operation in progress

Current device

Alpaca Focuser Sim -

127.0.0.1:32323

(Alpaca Focuser 0)

ASCOM Initiative

Welcome to Conform Universal!

Start

Stop

Conform Log

17:02:07.062

17:02:07.064 Properties

17:02:07.081 Absolute

17:02:07.095 IsMoving

17:02:07.109 MaxStep

17:02:07.127 MaxIncrement

17:02:07.138 Position

17:02:07.154 StepSize

17:02:07.164 TempCompAvailable

17:02:07.178 TempComp Read

17:02:07.188 TempComp Write

temperature compensation on

17:02:07.206 TempComp Write

temperature compensation off

17:02:07.235 Temperature

17:02:07.236

17:02:07.238 Methods

17:02:07.251 Halt

17:02:07.300 Move - TempComp False

45000

17:02:14.181 Move - TempComp False

completed

17:02:14.226 Move - TempComp False

17:02:14.237 Move - TempComp False

position: 50000

OK

True

False

50000

50000

50000

20

True

False

Successfully turned

OK

Successfully turned

OK

16.27

OK

Focuser halted OK

Moving to position:

Asynchronous move

OK

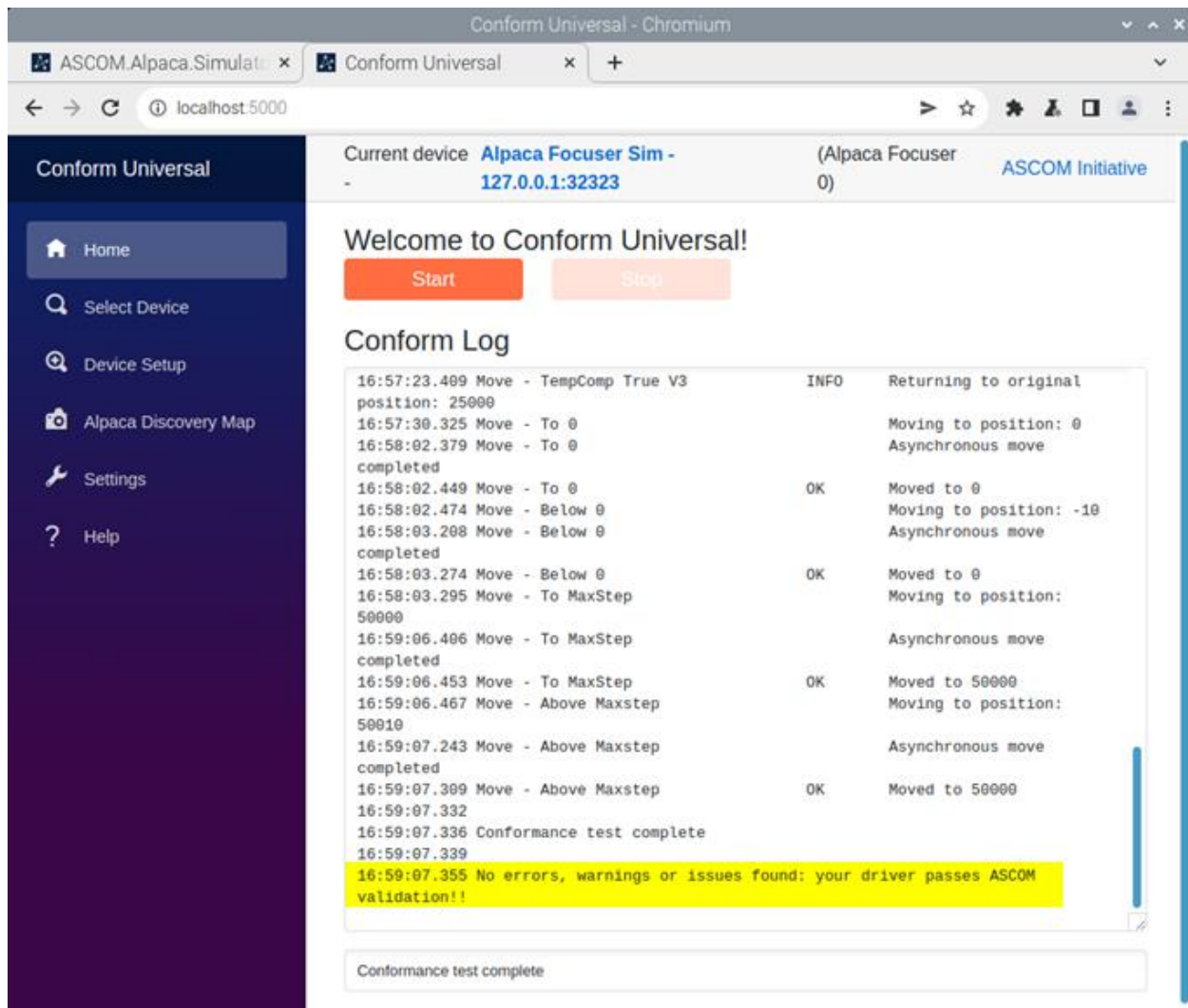
Absolute move OK

INFO

Returning to original

Focuser Move Returning to original position: 50000 Waiting for asynchronous move to complete, Position: 45000 / !

You will see the progress of the test, step by step. Depending on the device, it may take quite a while and cause many moves. When it is finished, you'll see something like this:



The log of the test is written to `/home/pi/ascom/logsyyyy-mm-dd/conform.report.txt`.

USING THE ALPACA PROTOCOL CHECKER

The Alpaca Protocol Checker validates your device's low-level HTTP and JSON for errors. Such errors include improper casing on endpoint URIs, missing or invalid required parameters, query string components, HTTP method usage, HTTP result codes, and more. If you are using one of the libraries supplied by the ASCOM Initiative, and are therefore working within the ASCOM Master API, you should not have problems with the low-level protocol. However, if you are rolling your own Alpaca JSON, programming your own HTTP requests (an application) or responding to HTTP requests (a device) this tool can help you "get it right".

Conform Universal - Chromium

ASCOM.Alpaca.Simulato x Conform Universal x +

localhost:41489/CheckAlpacaProtocol

ASCOM.Alpac...

Conform Universal

Current device - **Alpaca Focuser Sim - 127.0.0.1:32323** (Alpaca Focuser 0) [ASCOM Initiative](#)

Check Alpaca Protocol

Check Protocol **Stop**

Status: Congratulations, there were no errors, issues or information messages!

```

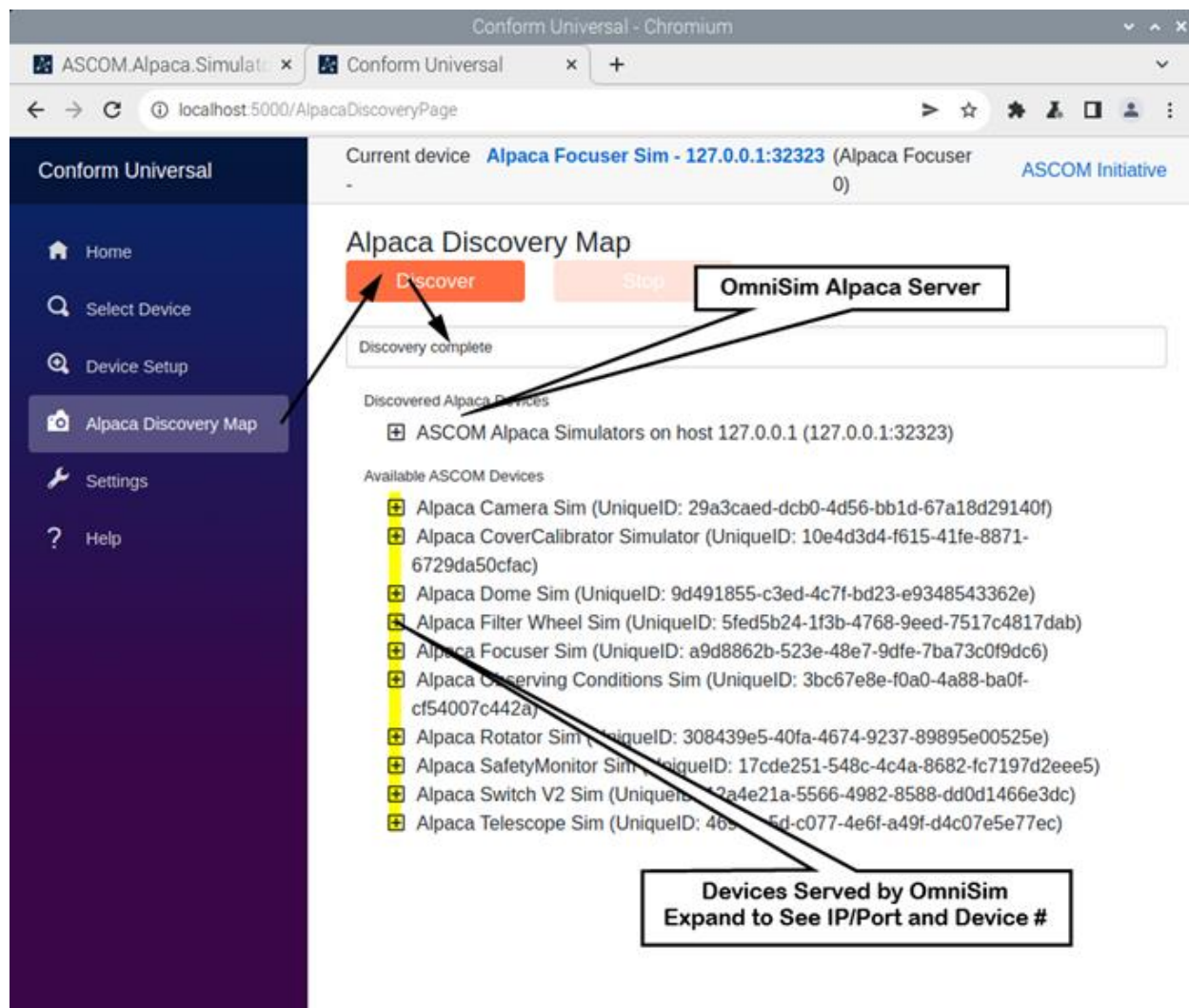
PUT Move OK Transaction Id negative - Received HTTP status 400
(BadRequest) as expected.
Response: "A data validation error occurred for
request /api/v1/focuser/0/move with error message <The value '-67890' is not valid.>. See
https://datatracker.ietf.org/doc/html/rfc7231#section-6.5.1."
PUT Move OK Transaction Id string - Received HTTP status 400
(BadRequest) as expected.
Response: "A data validation error occurred for
request /api/v1/focuser/0/move with error message <The value 'qweqwe' is not valid.>. See
https://datatracker.ietf.org/doc/html/rfc7231#section-6.5.1."
PUT Move OK Parameter Position (Good casing) - Received HTTP
status 200 (OK) as expected.
Response:
{"ClientTransactionID":67890,"ServerTransactionID":81,"ErrorNumber":0,"ErrorMessage":""}
PUT Connected OK False - Received HTTP status 200 (OK) as expected.
Response:
{"ClientTransactionID":67890,"ServerTransactionID":86,"ErrorNumber":0,"ErrorMessage":""}

No errors, issues or information alerts found: Your device passes ASCOM Alpaca protocol
validation!!

```

USING THE DISCOVERY MAPPING TOOL

The Alpaca Discovery Map tool is useful for finding out what devices are accessible from your Raspberry Pi. Open the Alpaca Discovery Map, click Discover, and see the results. Here's what you should see with the Workbench Omni Simulator running on your Raspberry Pi. Keep in mind that discovery operates only on your local network and not the public internet:



DISCOVERY DIAGNOSTICS

ConformU also includes a Discovery Diagnostics page (listed on the left-side list on later versions). When invoked, it provides information to support the discovery process and correctly implementing the Alpaca management API. Discovery Diagnostics confirms correct operation and reports issues such as:

- Incorrect JSON discovery response casing or structure
- Incorrect supported API version numbers
- Missing or empty manufacturer name and version fields
- Incorrectly spelled ASCOM device types
- Missing or empty unique ID

EXAMINING HTTP/REST MESSAGES WITH WIRESHARK

While familiarizing yourself with Alpaca, as well as testing your app or driver, it can be very useful to see the actual HTTP/REST messages between the app (requestor) and the driver (responder). By installing the amazing [Wireshark](#) on your Pi along with the OmniSim, you can see the actual HTTP/REST messages. It's easy to install but being so

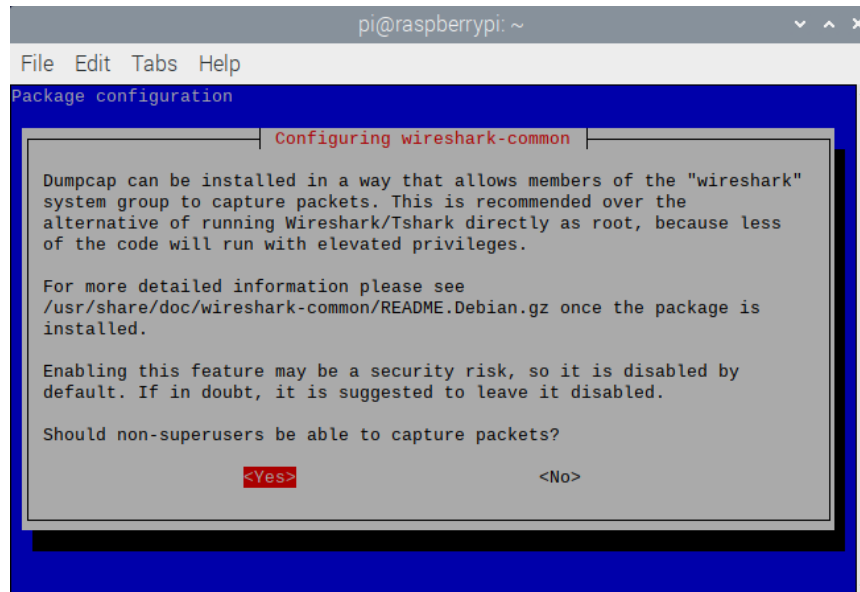
powerful it will take you some time to get used to it. To save you time, here is a link to the [PDF Wireshark User Manual](#).

INSTALLING WIRESHARK AND SETTING PRIVILEGES

To install it on the Pi, you need about 100MB. In a shell

```
pi@raspberrypi:~ $sudo apt install wireshark
```

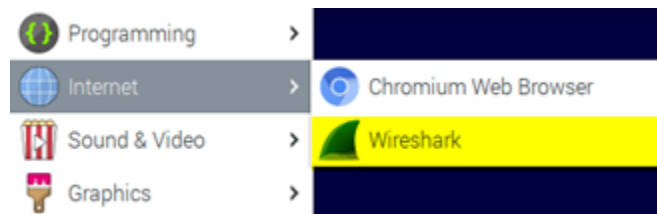
During installation you'll see this



Make sure to answer **<Yes>** to this so you don't have to start Wireshark with root privs. But there is more, note it says you still need to be a member of the "wireshark" group. Once the installation completes, add yourself (typically you are user "pi"):

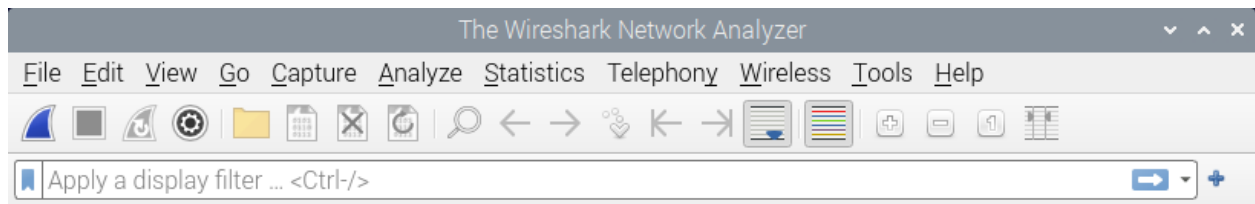
```
pi@raspberrypi:~ $sudo usermod -a -G wireshark pi
```

Now log out and back in to join the "wireshark" group. You will find Wireshark in the Berry menu under Internet.



CHECKING THE WIRESHARK INSTALLATION

Now start Wireshark and be sure you see the network interfaces, indicating that you have successfully allowed non-root capturing and joined the "wireshark" group. You should see this:



Welcome to Wireshark

Capture

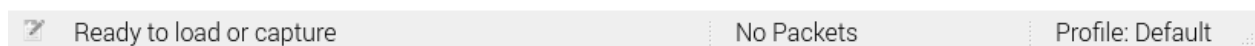
...using this filter: All interfaces shown



Learn

[User's Guide](#) · [Wiki](#) · [Questions and Answers](#) · [Mailing Lists](#)

You are running Wireshark 3.4.10 (Git v3.4.10 packaged as 3.4.10-0+deb11u1).



If you see wlan0 and traffic, then you're ready to use Wireshark. If you are on Ethernet, the traffic will be on eth0. Otherwise:

1. Did you answer Yes to the non-root capture? You can check by entering this command `pi@raspberrypi:~ $sudo dpkg-reconfigure wireshark-common` which will show the allow non-root dialog that appeared during installation. Answer **<Yes>**.
2. Did you log out and back in after adding yourself to the "wireshark" group?

Go back and repeat the installation steps till you see a Wireshark window with the physical interfaces as shown above.

SETTING UP A TEST & LEARNING ENVIRONMENT

You can use an application such as the Windows DeviceHub (via an Alpaca dynamic driver as shown above), [SkySafari Plus or Pro](#) on iOS or [Cartes du Ciel](#) on any of its supported platforms to connect to the OmniSim's Telescope Simulator. Or you can use the Workbench's ConformU tool though it produces a large amount of traffic. Once you have an app (anywhere) talking to the simulated telescope on the Pi, you can use the amazing Wireshark to see the HTTP/REST traffic between them. Here we use SkySafari Pro on 192.168.0.21, and we're running Wireshark on the same Pi that's running the OmniSimulator, on 192.168.0.42. Have a look at this packet capture of SkySafari's initial connect to the Telescope. The key to setting this up is the display filter, which limits the display to HTTP and port 32323 (the OmniSim's port). If you make a mistake in the display filter the background will turn reddish. Without the display filter you will see a lot of uninteresting (for our purposes) trash.

Initial request - What devices do you offer?

Display Filter: http and tcp.port == 32323

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-----------|--------------|--------------|-----------|--------|---|
| 207 | 51.534527 | 192.168.0.21 | 192.168.0.42 | HTTP | 233 | GET /management/v1/configureddevices?ClientID=67502 |
| 210 | 51.537991 | 192.168.0.42 | 192.168.0.21 | HTTP/JSON | 201 | HTTP/1.1 200 OK, JavaScript Object Notation (appli |
| 218 | 51.549786 | 192.168.0.21 | 192.168.0.42 | HTTP | 313 | PUT /api/v1/telescope/0/connected HTTP/1.1 (applic |
| 220 | 51.557091 | 192.168.0.42 | 192.168.0.21 | HTTP/JSON | 313 | HTTP/1.1 200 OK, JavaScript Object Notation (appli |
| 228 | 51.57113 | 192.168.0.21 | 192.168.0.42 | HTTP | 233 | GET /api/v1/telescope/0/canslewasync?ClientID=67502 |
| 238 | 51.61166 | 192.168.0.42 | 192.168.0.21 | HTTP/JSON | 71 | HTTP/1.1 200 OK, JavaScript Object Notation (appli |
| | | 192.168.0.21 | 192.168.0.42 | HTTP | 228 | GET /api/v1/telescope/0/cansync?ClientID=675023106& |
| | | 192.168.0.42 | 192.168.0.21 | HTTP/JSON | 326 | HTTP/1.1 200 OK, JavaScript Object Notation (appli |
| | | 192.168.0.21 | 192.168.0.42 | HTTP | 228 | GET /api/v1/telescope/0/canpark?ClientID=675023106& |
| | | 192.168.0.42 | 192.168.0.21 | HTTP/JSON | 326 | HTTP/1.1 200 OK, JavaScript Object Notation (appli |
| | | 192.168.0.21 | 192.168.0.42 | HTTP | 235 | GET /api/v1/telescope/0/cansettracking?ClientID=675 |
| | | 192.168.0.42 | 192.168.0.21 | HTTP/JSON | 326 | HTTP/1.1 200 OK, JavaScript Object Notation (appli |
| | | 192.168.0.21 | 192.168.0.42 | HTTP | 239 | GET /api/v1/telescope/0/canmoveaxis?Axis=0&ClientID |

Frame 210: 201 bytes on wire (1608 bits), 201 bytes captured (1608 bits) on interface wlan0, id 0

Ethernet II, Src: Raspberr_21:8a:95 (dc:a6:32:21:8a:95), Dst: Apple_75:c5:42 (88:66:5a:75:c5:42)

Internet Protocol Version 4, Src: 192.168.0.42, Dst: 192.168.0.21

Transmission Control Protocol, Src Port: 32323, Dst Port: 52200, Seq: 1449, Ack: 168, Len: 135

[2 Reassembled TCP Segments (1563 bytes): #209(1448), #210(135)]

Hypertext Transfer Protocol

JavaScript Object Notation: application/json

- Object
 - Member Key: Value
 - Array
 - Object
 - Member Key: DeviceName
 - String value: Alpaca Camera Sim
 - Key: DeviceName
 - Member Key: DeviceType
 - String value: Camera
 - Key: DeviceType
 - Member Key: DeviceNumber
 - Number value: 0
 - Key: DeviceNumber
 - Member Key: UniqueID
 - String value: cd623cff-4d60-43be-a152-f0df3655619f
 - Key: UniqueID
 - Object
 - Member Key: DeviceName
 - String value: Alpaca CoverCalibrator Simulator

wireshark_wlan0AFRAG1.pcapng Packets: 1328 · Displayed: 62 (4.7%) · Dropped: 0 (0.0%) Profile: Default

Look at the list of REST transactions. The first gets the list of devices as shown. Next you see a PUT of **true** to the telescope's **connected** endpoint, and this succeeds. Then it GETs some capability properties: **canslewasync**, **cansync**, **canpark**, etc.

It's beyond the scope of this document to be a tutorial on Wireshark. There are loads of videos on YouTube covering Wireshark. And there's always the [PDF Wireshark User Manual](#).