

ALPACA WORKBENCH FOR RASPBERRY PI 10 (BUSTER) AND 11 (BULLDOG)

Bob Denny (January 2022)

Contact on [ASCOM Talk Developer Forum](#)

This document shows you how to build an Alpaca workbench consisting of the ASCOM Alpaca "Omni Simulators" package along with [Wireshark](#) to analyze the HTTP/REST network data. Since this is (primarily) a developer tool, help will be provided on the [ASCOM Driver and Application Development Forum](#).

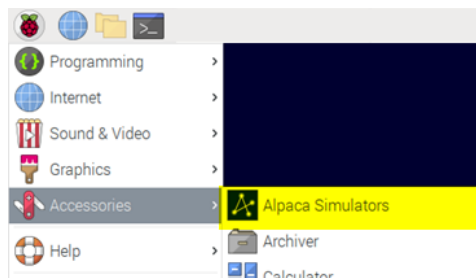
CONTENTS

INSTALLATION	2
INITIAL START AND CONFIGURATION OF SIMULATORS	3
Setting up the Host App (Server)	3
Setting up Devices	5
ROUTINE USE - ALPACA	6
ROUTINE USE - FROM WINDOWS APPS VIA A DYNAMIC DRIVER	9
USAGE FROM WINDOWS APPS THAT DON'T USE THE CHOOSER (SGP, NINA, ETC).....	11
Chooser in ASCOM Diagnostics	11
ADVANCED ALPACA USAGE FROM WINDOWS APPS.....	12
EXPLORING ALPACA AND THE API WITH THE OPENAPI BROWSER	13
EXAMINING HTTP/REST MESSAGES WITH WIRESHARK	15
Installing Wireshark and Setting Privileges.....	15
Checking the Wireshark Installation	16
Setting up a Test & Learning Environment	16

INSTALLATION

The Omni Simulators components placed into **/opt¹/ascom.alpaca.simulators** along with the .desktop file needed in **/usr/share/applications** for the simulators app to appear in the Desktop GUI's "berry" menu under Accessories.

The files and folders *within* the **opt** and **usr** folders go into the corresponding places under **/opt** and **/usr** on your Buster or Bulldog OS. Of course you will need to do this with root privileges so use the shell/terminal and Linux commands to do the copying. When done, make sure the **/opt/ascom.alpaca.simulators** directory exists and shows a bunch of files plus a **wwwroot** subfolder. Also make sure that the file **/usr/share/applications/ascom.alpaca.simulators/desktop** exists. You may see the berry menu flash as you add the .desktop file, but you may need to log out/in or reboot. In the end, the berry menu should show Alpaca Simulators:



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

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, several harmless GUI and other error messages. Shortly thereafter your default browser (normally Chromium) will appear.



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 of the simulators share a single host app (server) and the common settings for this are here.

ASCOM.Alpaca.Simulators

- Camera
- CoverCalibrator
- Dome
- Filter Wheel
- Focuser
- ObservingConditions
- Rotator
- SafetyMonitor
- Switch
- Telescope
- Driver Setup**
- About

Server Settings

Server Version: 0.1.1
 Server Location:
 Allow Remote Access: ☒
 Server (Alpaca) Port:
 Open Browser on Start: ☒

Save

Bound Addresses

Addresses server is listening on.

127.0.0.1
 ::1
 192.168.0.40
 fe80::2d0:ba7b:a6ac:2478%3

Driver Settings

Prevent Disconnects: ☐

Save

Alpaca Discovery Settings

Allow Discovery: ☒
 Discovery Port:
 Respond to localhost on loopback only: ☒

Save

Respond to localhost on loopback only: ☒

Save

Security Settings

Require Authorization: ☐
 User Name:
 Password:
 Repeat:

Save

Log Settings

The logging system will log out at or above the specified level.

Log Level: Save

Run OpenAPI / Swagger

The OpenAPI / Swagger integration allows for the Alpaca API documentation to be shown by this server. It also allows calls to be made directly in the browser. Note that this will apply on the next time this driver runs.

Run OpenAPI: ☒ Save

Shutdown Driver

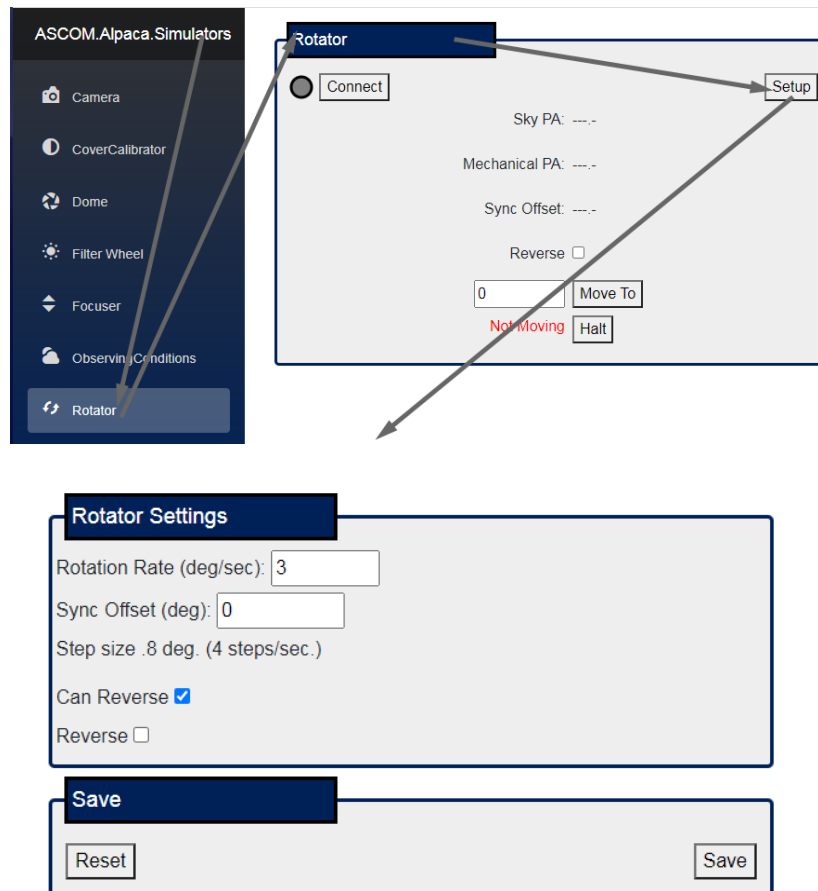
Shutdown:

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:

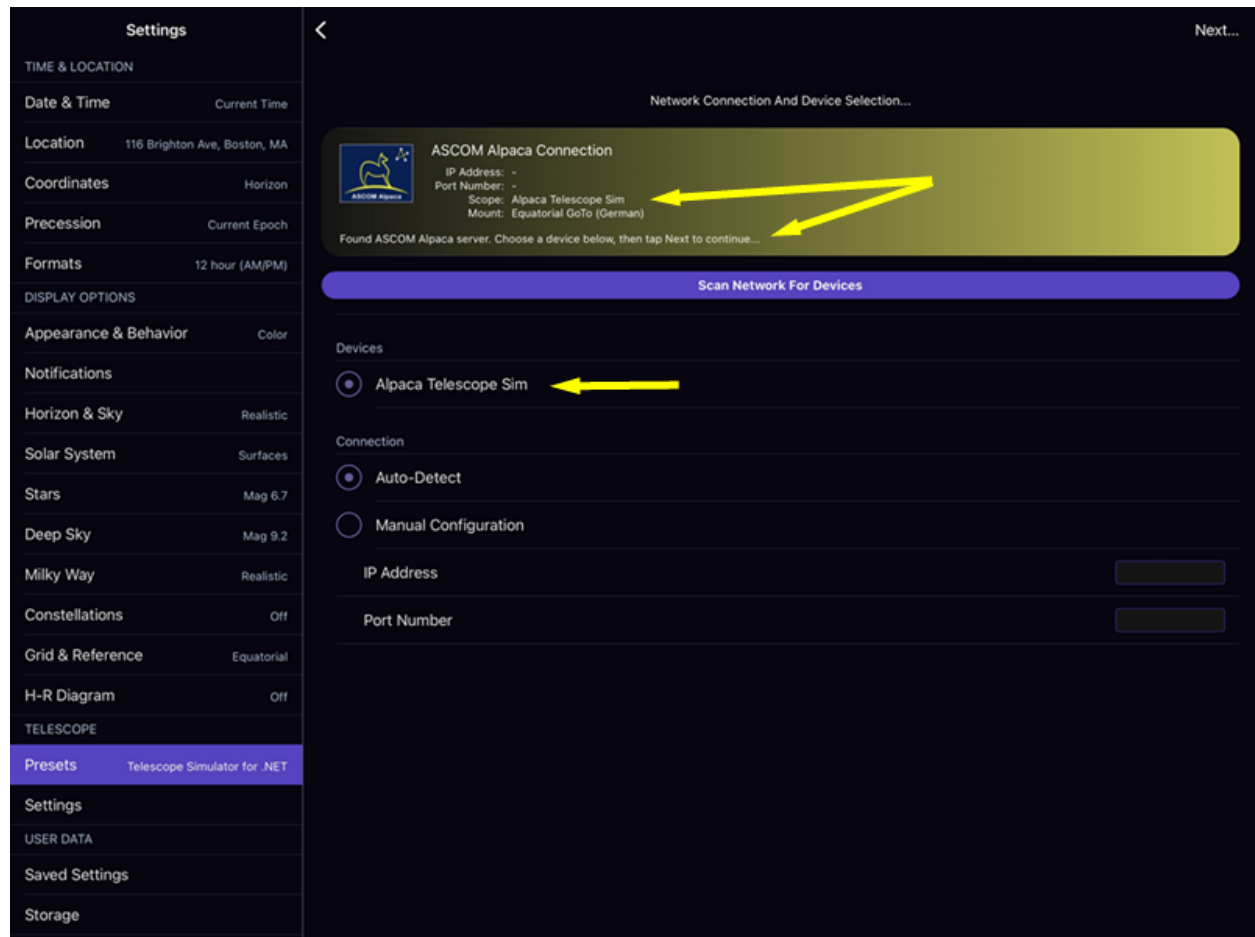


Clicking Save makes your changes persistent. Use the browser's back arrow to close the setup dialog. Hopefully this will get you going with the simulators you want to use.

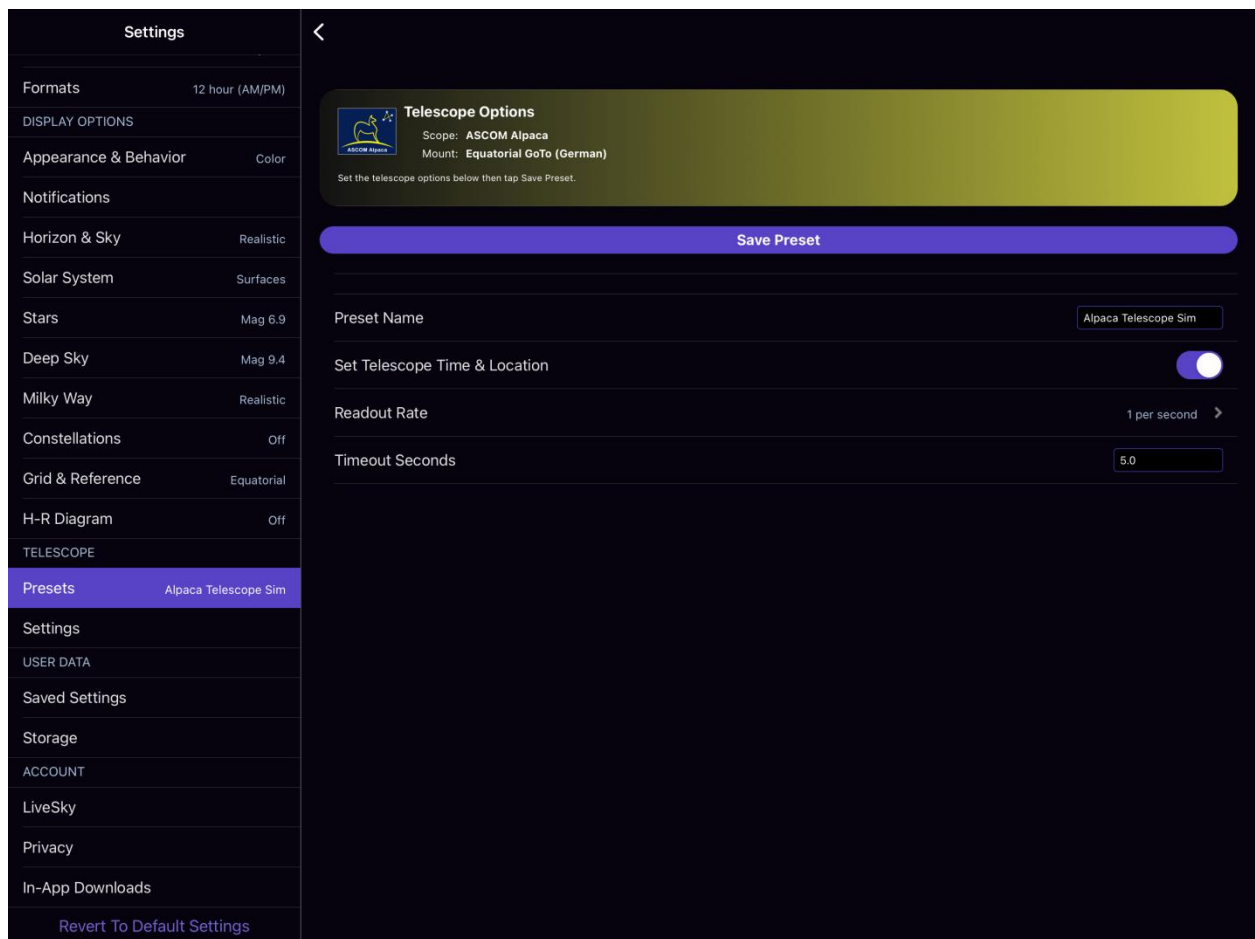
ROUTINE USE - ALPACA

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

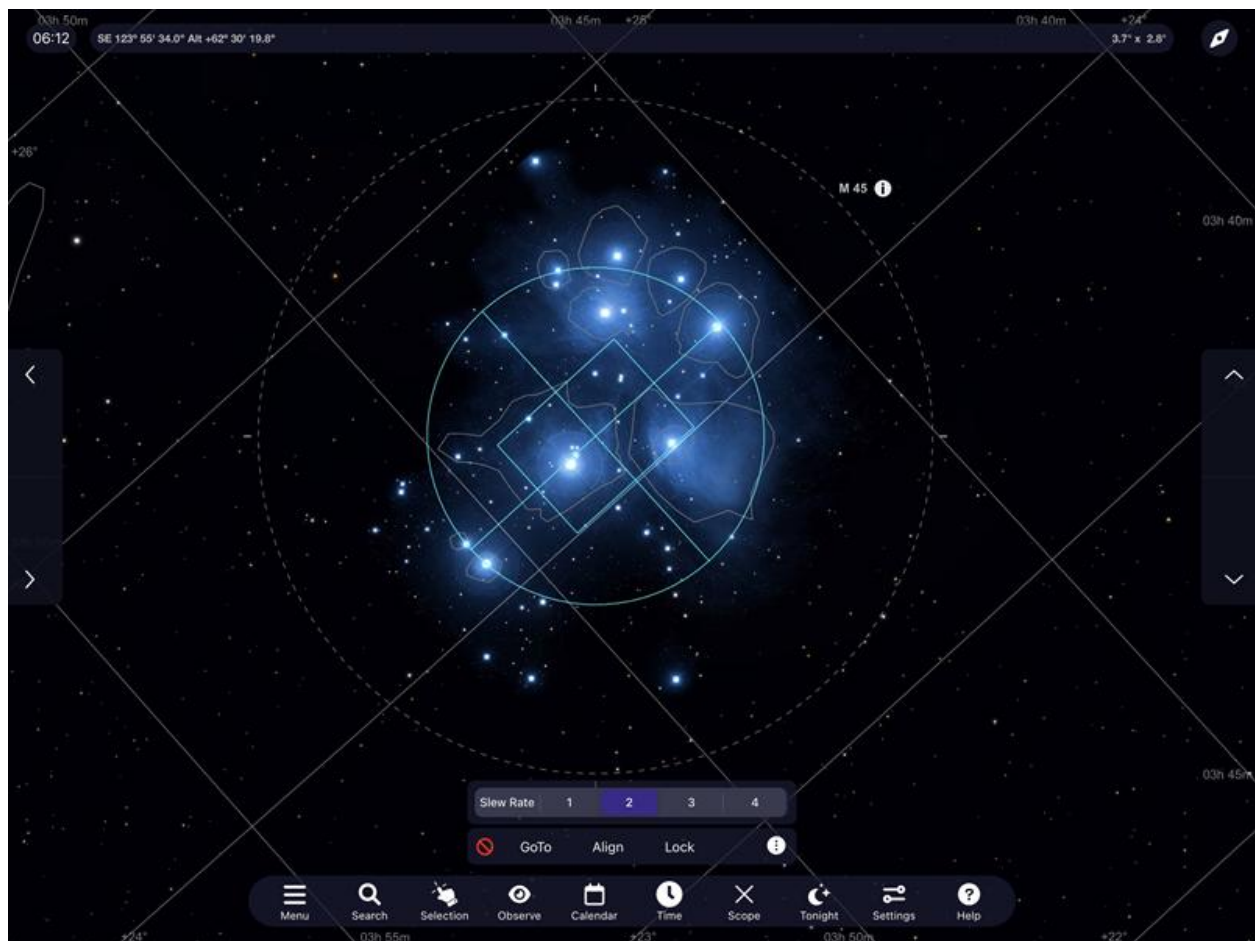
Once the simulators are running, you can close the browser and the simulators should simply answer and provide the simulated devices to any app that wants to use them. For example, using [SkySafari 7 Plus or Pro](#), make a new Telescope Preset, select ASCOM Alpaca as the Connection Type. Leave Auto-Detect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

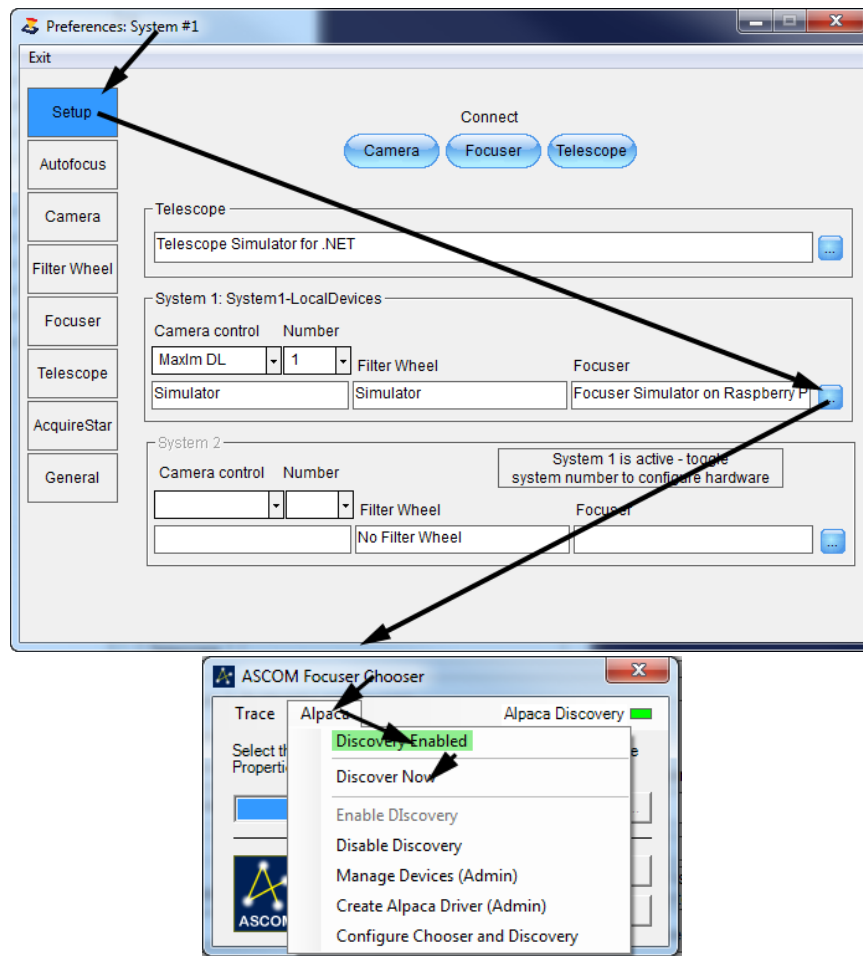


ROUTINE USE - FROM WINDOWS APPS VIA A DYNAMIC DRIVER

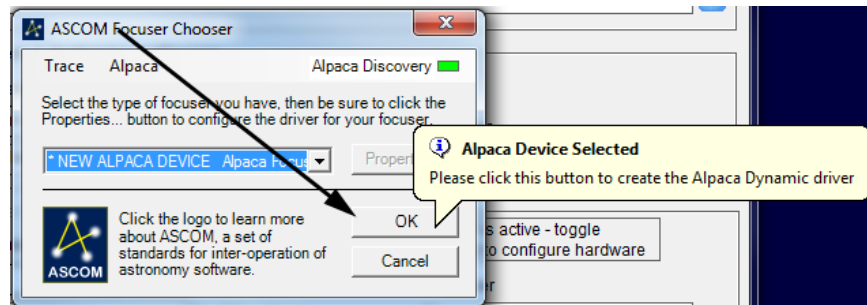
Please note: Windows is not required for Alpaca as we saw above.

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**.

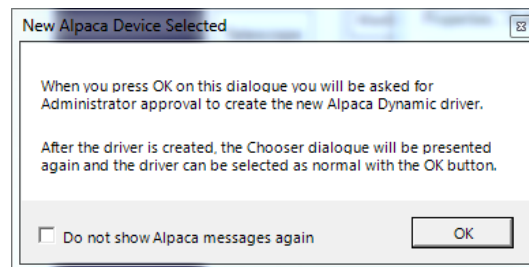
You only need to do this once for this Alpaca device. In your ASCOM-compatible astronomy app, use the Chooser as usual to specify the type of device. Let's say you are using FocusMax and your focuser is a self-contained WiFi Alpaca device. You can use the Omni Simulator's WiFi Alpaca focuser to simulate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



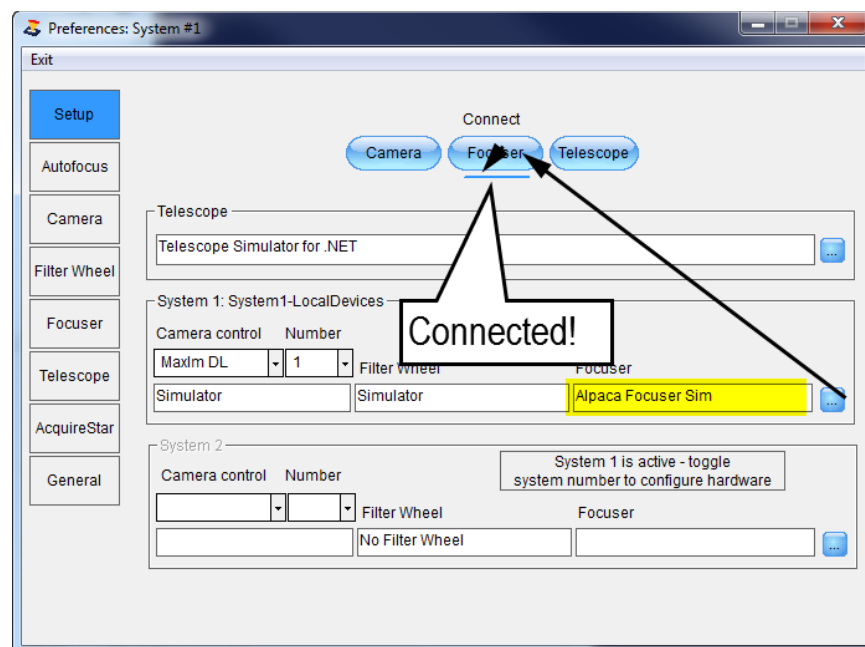
Enable Discovery (one time setting) then Discover Now. The Chooser will automatically discover when opened the next time and thereafter. You will see the Alpaca Discovery light flash on and off. Once it is finished, click the device list as usual, you will see a new Alpaca-discovered Focuser listed (it's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



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

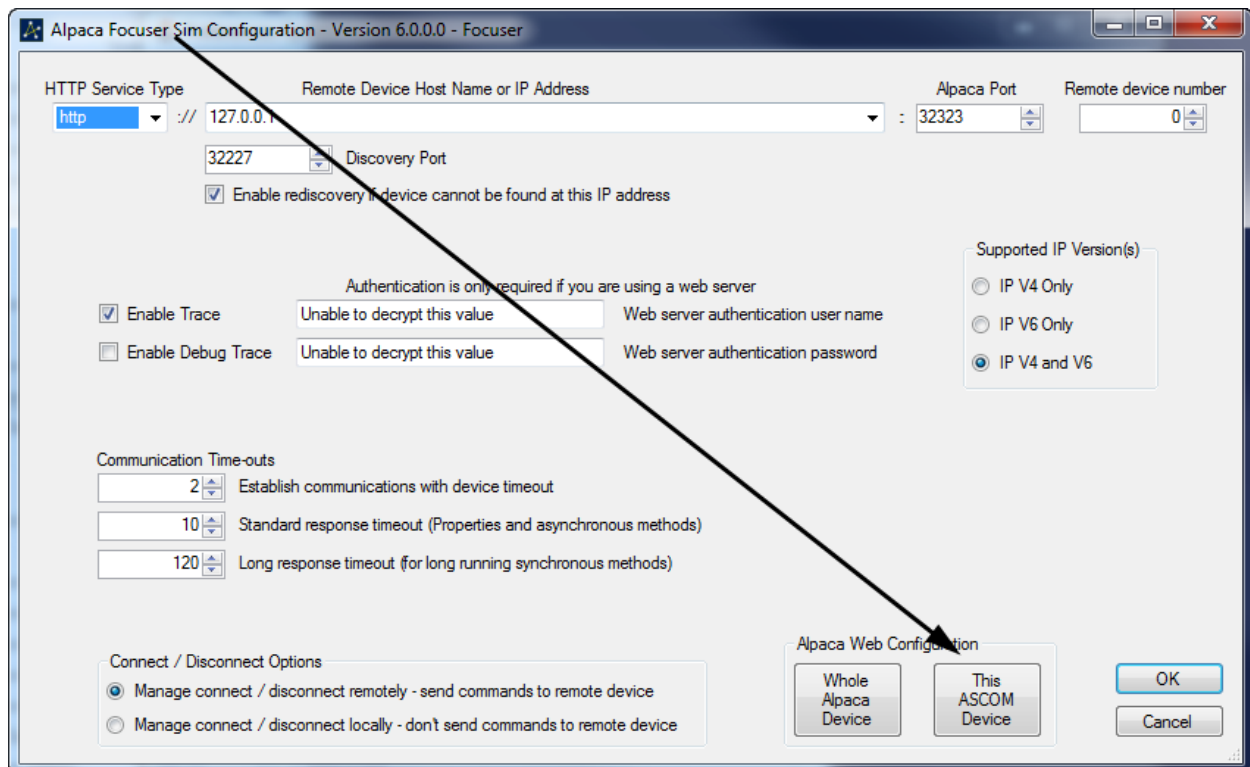
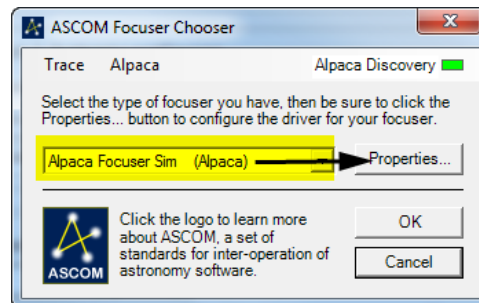
Programs such as Sequence Generator Pro (SGP) and Nighttime Imaging 'N' Astronomy (NINA) don't use the ASCOM Chooser to select their ASCOM devices. Therefore they do not have in-app access to the automatic Alpaca gateway driver creation feature in the ASCOM Chooser. This is no problem though. You can 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

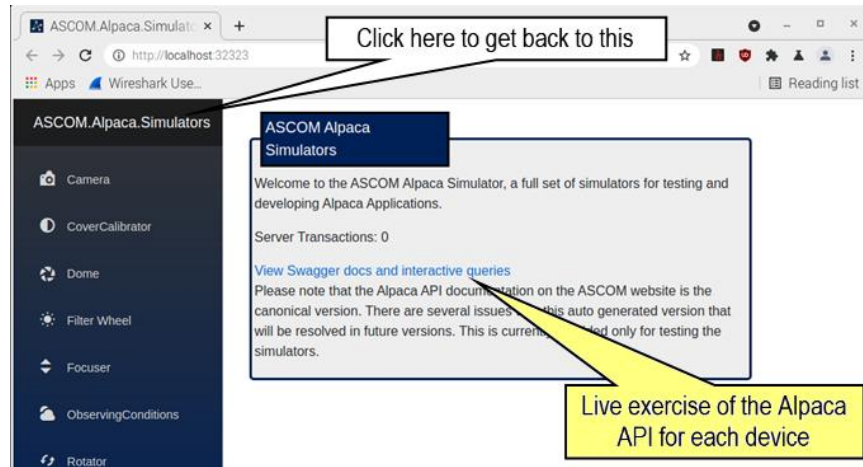
Any time you open the Chooser and one of the Alpaca drivers is selected, you can click **Properties...** to show a window where you can control advanced network capabilities for your app, including username/password security, ports, etc.



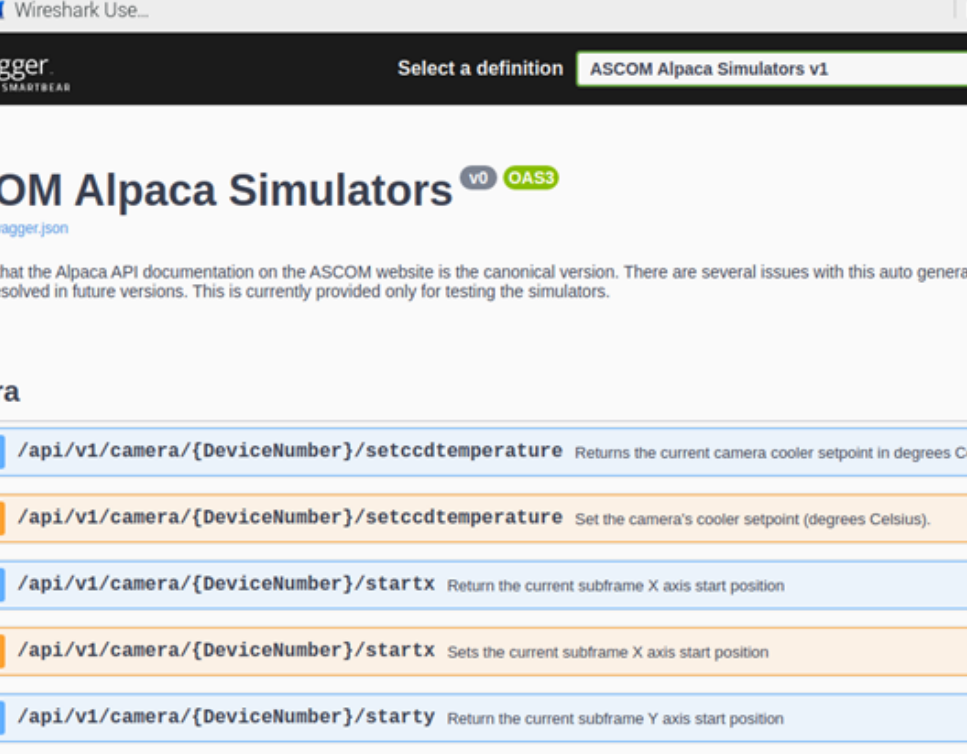
If you are network-skilled most of this will make sense. If not don't worry about it. But there is one cool thing you can do from here. If you click "This ASCOM Device", you'll see the device (Focuser in this case) setup page in the browser. Try it. Once you see this and understand what this does, click 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 AND THE API WITH THE OPENAPI BROWSER

The OmniSimulator includes a complete implementation of the [OpenAPI Browser \(formerly known as Swagger UI\)](#). All of the Alpaca device APIs are presented as visual documentation, including live "Try it out" capability. To get there, return 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.



ASCOM Alpaca Simulators v1

Select a definition

ASCOM Alpaca Simulators v1

ASCOM Alpaca Simulators v0 OAS3

/swagger/v1/swagger.json

Please note that the Alpaca API documentation on the ASCOM website is the canonical version. There are several issues with this auto generated version that will be resolved in future versions. This is currently provided only for testing the simulators.

Camera

- GET** `/api/v1/camera/{DeviceNumber}/setccdtemperature` Returns the current camera cooler setpoint in degrees Celsius.
- PUT** `/api/v1/camera/{DeviceNumber}/setccdtemperature` Set the camera's cooler setpoint (degrees Celsius).
- GET** `/api/v1/camera/{DeviceNumber}/startx` Return the current subframe X axis start position
- PUT** `/api/v1/camera/{DeviceNumber}/startx` Sets the current subframe X axis start position
- GET** `/api/v1/camera/{DeviceNumber}/starty` Return the current subframe Y axis start position
- PUT** `/api/v1/camera/{DeviceNumber}/starty` Sets the current subframe Y axis start position
- GET** `/api/v1/camera/{DeviceNumber}/subexposureduration` Camera's sub-exposure interval
- PUT** `/api/v1/camera/{DeviceNumber}/subexposureduration` Sets the current Sub Exposure Duration
- PUT** `/api/v1/camera/{DeviceNumber}/abortexposure` Aborts the current exposure

EXAMINING HTTP/REST MESSAGES WITH WIRESHARK

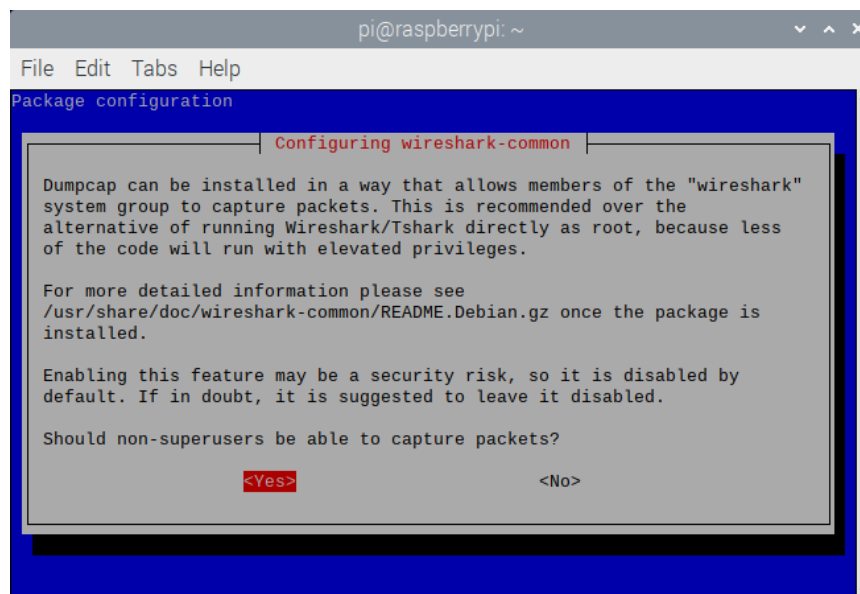
While familiarizing yourself with Alpaca, as well as testing your app or driver, it can be very useful to see the actual HTTP/REST messages between the app (requestor) and the driver (responder). By installing the amazing [Wireshark](#) on your Pi along with the OmniSim, you can see the actual HTTP/REST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

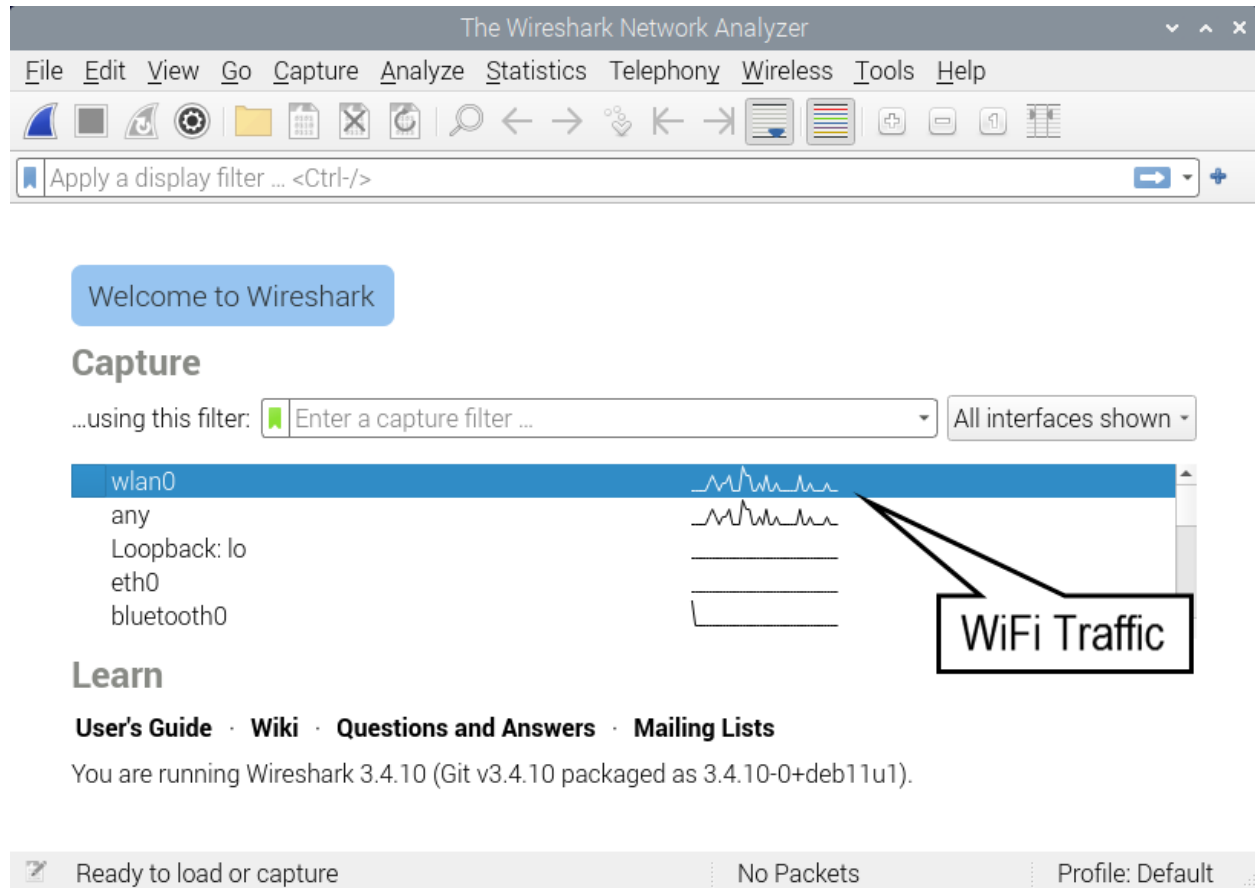
```
pi@raspberrypi:~ $sudo usermod -a -G wireshark pi
```

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:



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. Then once you have an app 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

Response: Object Array.
First is the camera sim.

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	51.549086	192.168.0.21	192.168.0.42	HTTP	313	PUT /api/v1/telescope/0/connected HTTP/1.1 (applic
220	51.55091	192.168.0.42	192.168.0.21	HTTP/JSON	313	HTTP/1.1 200 OK, JavaScript Object Notation (appli
228	51.5513	192.168.0.21	192.168.0.42	HTTP	233	GET /api/v1/telescope/0/canslewasync?ClientID=67502
238	51.5516	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
269	51.609229438	192.168.0.21	192.168.0.42	HTTP	235	GET /api/v1/telescope/0/cansettracking?ClientID=675
271	51.611437343	192.168.0.42	192.168.0.21	HTTP/JSON	326	HTTP/1.1 200 OK, JavaScript Object Notation (appli
270	51.62004508	192.168.0.21	192.168.0.42	HTTP	230	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: Aloaca 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](#).