ASCOM REMOTE

Installation and Configuration

Abstract

Instructions for installing and configuring ASCOM Remote

Peter Simpson
peter@peterandjill.co.uk

Contents

1. ASCOM Remote Installation and Setup

1.1 Pre-Requisites

Please note that ASCOM Remote requires .NET Framework 4.6.2, which means that the operating system must be Windows 7 SP1 or later because .NET 4.6.2 is not available on earlier operating systems.

1.2 Installation

The installer provides options to install either the Remote Clients, the Remote Server or both. The installer will:

- **Remote Clients**: Install new 1 client device driver for each device type e.g. ASCOM.Remote1.Telescope, which will appear as ASCOM Remote Client 1 in Chooser.
- **Remote Server:** Install the remote driver server in your Start/ ASCOM Remote folder.
- Configure firewall permissions for the clients and or server.

2. Configuring the number of remote clients

After installation there will be one remote client of each device type, but you can configure the number of clients in each device type through the "Remote Client Configuration" utility that will be found in your Start / ASCOM Remote folder. This is to support complex configurations where there may be multiple devices of same type, such as cameras, focusers and filter wheels.

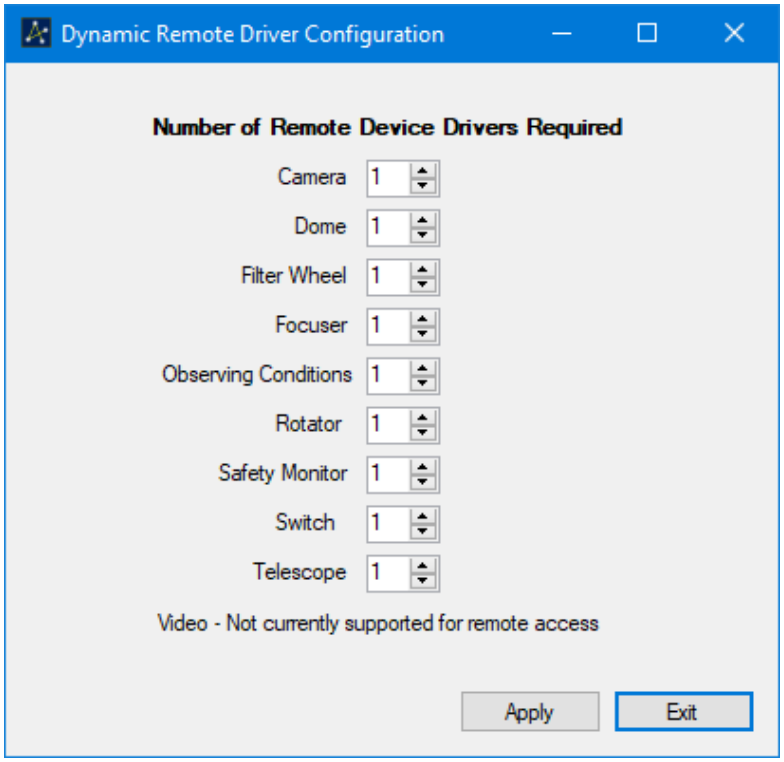


Figure 1 - Utility to configure the number of drivers of each device type

These will appear as normal ASCOM drivers named ASCOM Remote Client 1, ASCOM Remote Client 2 etc. and can be selected and configured for each application through Chooser in the normal way.

The remote clients are all hubs in their own right and can support connections from multiple clients.

3. Configuring Remote Clients

The Remote Clients appear in Chooser as normal ASCOM drivers:

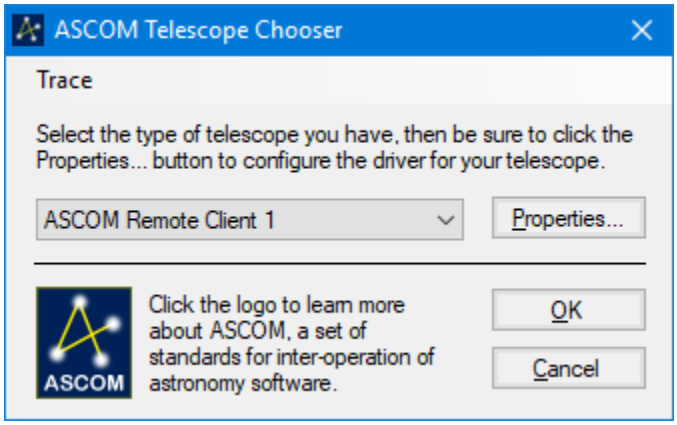


Figure 2 - Telescope Chooser showing a remote client

and can be configured through Chooser's Properties button in the usual way.

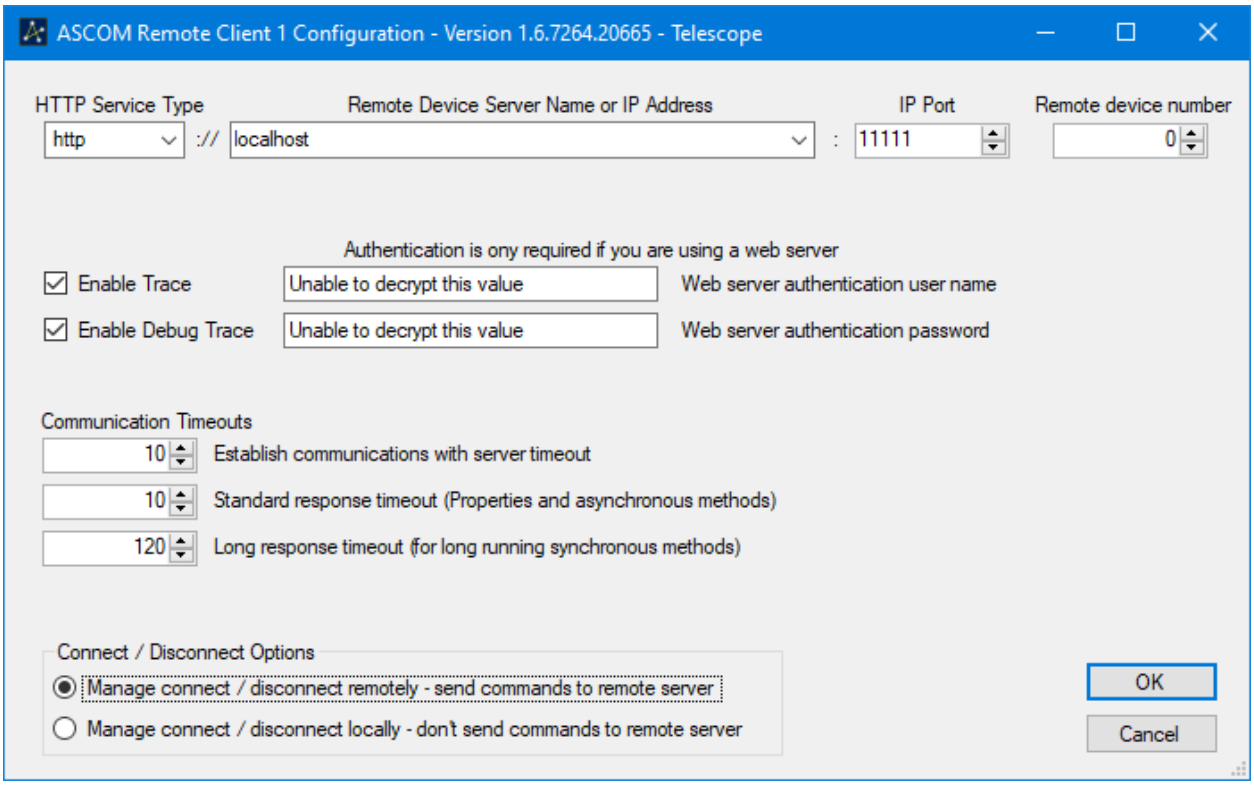


Figure 3 - Remote client configuration form

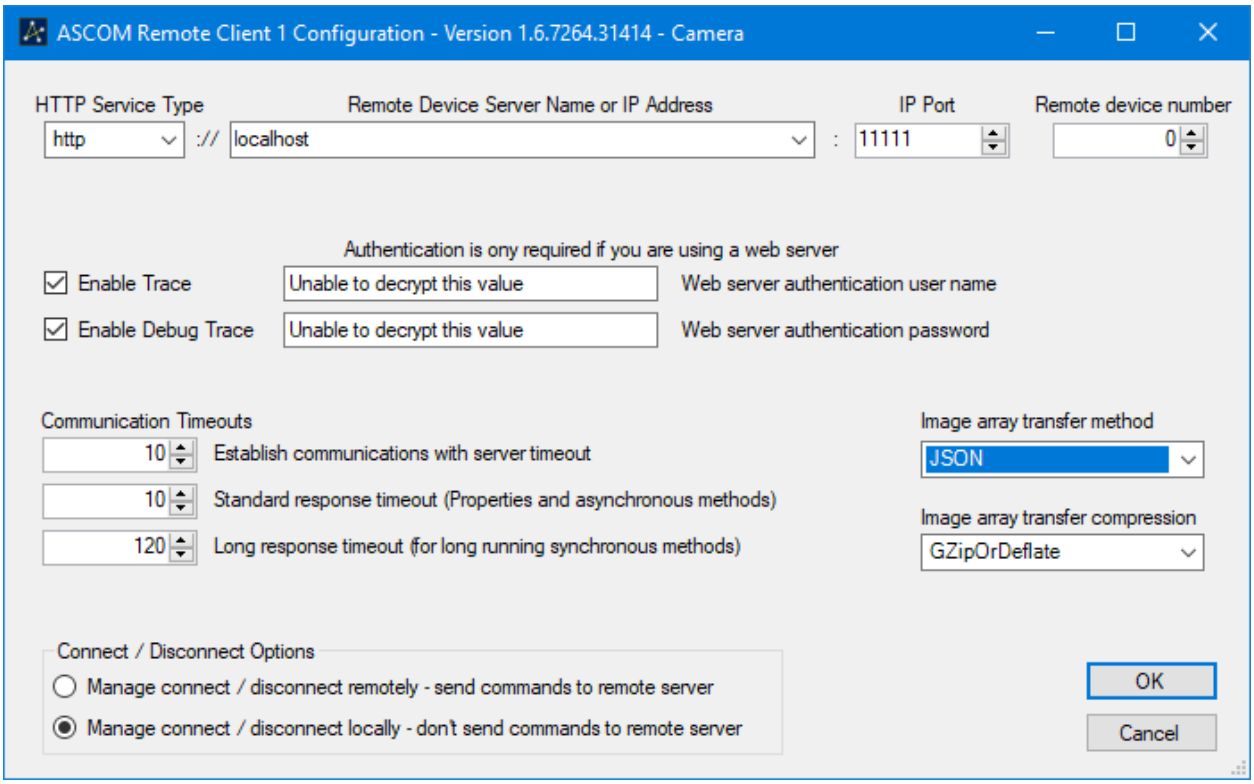
The service type (HTTP/HTTPS), IP address or host name and port set on the configuration form must match the values used when configuring the Remote Server.

There are three communication timeouts, one for establishing an initial connection with the Remote Server the second for relatively quick response commands such as CanXXX properties and a third for slow response commands such as Telescope.SlewToCoordinates. The standard response timeout

default should suit most requirements, but you may need to increase the slow response timeout depending on the longest command completion time expected under normal circumstances for your remote device.

3.1 Camera Device Configuration

Camera devices support two additional configuration options that determine aspects of image array transfer.



3.1.1 Image Array Transfer Method

- **JSON** - Uses the original JSON encoding mechanic per the Alpaca specification, which can be slow for large images.
- **Base64HandOff** - Requests use of a base64 handoff mechanic, which returns a small JSON response (see section 4.3) and permits downloading of a base64 encoded version of the image. In testing transfer times for 4000 x 3000 images reduced from typically 12 seconds to less than 2 seconds. Transfer compression cannot be requested in this mode because it always degraded overall timings.

3.1.2 Image Array Transfer Compression

- None - No compression will be requested
- Deflate - Deflate compression will be requested
- GZip - GZip compression will be requested
- GZipOrDeflate - Both GZip and Deflate compression will be requested, the remote device will choose which is used.

3.2 Authentication

The username and password fields allow authentication credentials for the remote server to be configured. Any values entered are encrypted before being persisted in the Profile. These fields are only useful if Apache or some other web server is used to proxy incoming remote device server connections and it has been configured to require a password to access the remote server URIs.

Notes

- For testing its fine to run the driver and the remote server on the same PC and to use 127.0.0.1 as the IP address for both clients and server
- The remote server itself only supports HTTP, see Section **Error! Reference source not found.**

4. ASCOM Remote Server

The Remote Server application start shortcut is located in the Start Menu / ASCOM Remote folder.

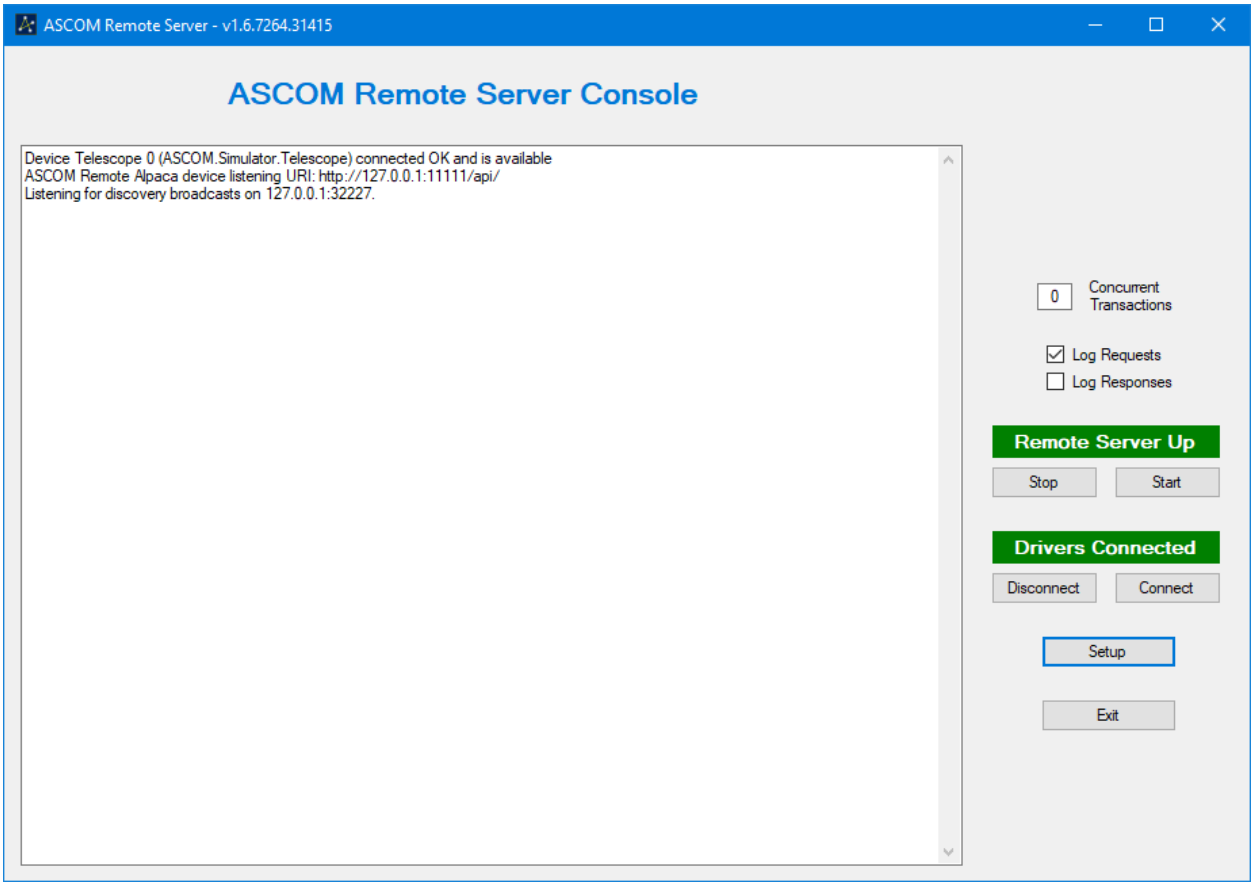


Figure 4 - Remote server console

When the Remote Server starts it will list:

- The configured devices
- The URI on which it is listening for requests
- The Alpaca discovery port on which it is listening

Remote server configuration is effected through the Setup button.

4.1 Setup - Device Configuration

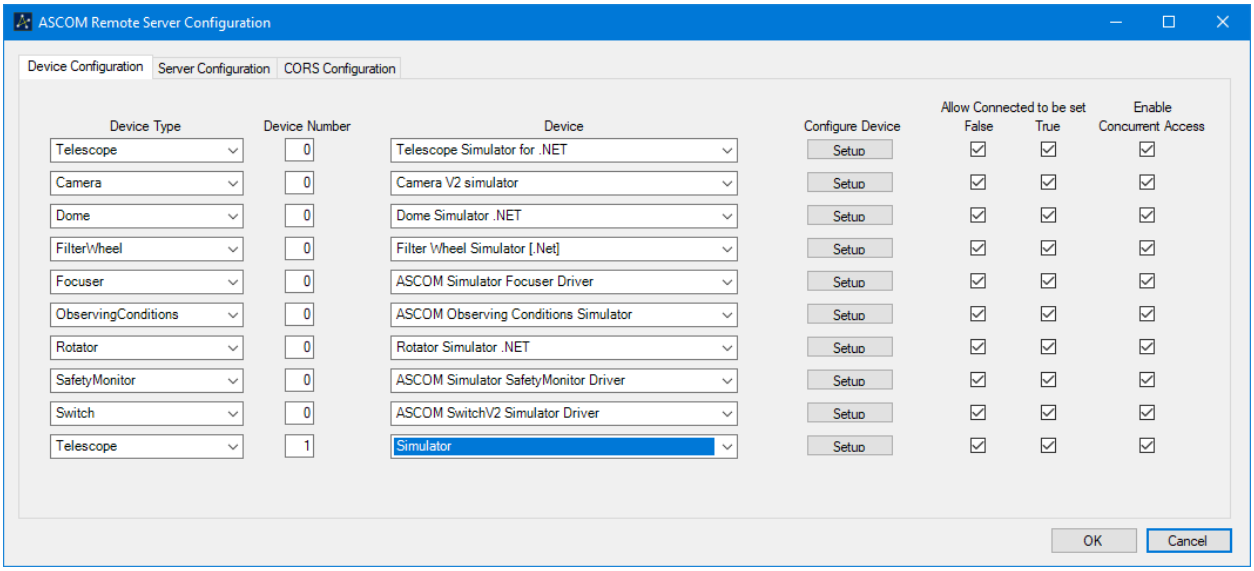


Figure 5 - Device configuration dialogue

4.1.1 Device Selection

To set up a device to be remotely served, first select the type of device in one of the "Device Type" drop-down boxes, then select its driver from the corresponding "Device" drop-down box. Make sure that all unused "Device Type" dropdowns are set to "None".

"Device Numbers" are automatically assigned as device types are selected and relate to the number of devices of that specific device type that are configured. E.g. the first focuser driver that is configured will be focuser device "0" while the second focuser device will be focuser device "1" etc.

The configured "Device Number" and "Device Type" uniquely identifies a remote device and **it is these that must be configured in the remote client** to specify the required remote device.

4.1.2 Device Configuration

The device's configuration screen can be accessed through its "Setup" button.

4.1.3 Connected State Management

The "Allow Connected" check boxes determine whether "Set Connected True" and "Set Connected False" requests will be sent to the device, which enables a device to be maintained in a connected state even if a client disconnects. When the "Connected" check boxes are unset, client drivers will see Telescope.Connected changing state as they expect, but the state of the remote device will not change.

This feature will be of value in multi-client environments where the observatory operator can prevent devices being commanded offline by one client while still in use by another.

4.1.4 Enable Concurrent Access

These check boxes control whether the Remote Server will send commands to each device as they are received, even if previous commands have not completed, or whether the Remote Server will queue the commands and send them to the device one at a time, in the order in which they were received.

4.2 Setup - Server Configuration

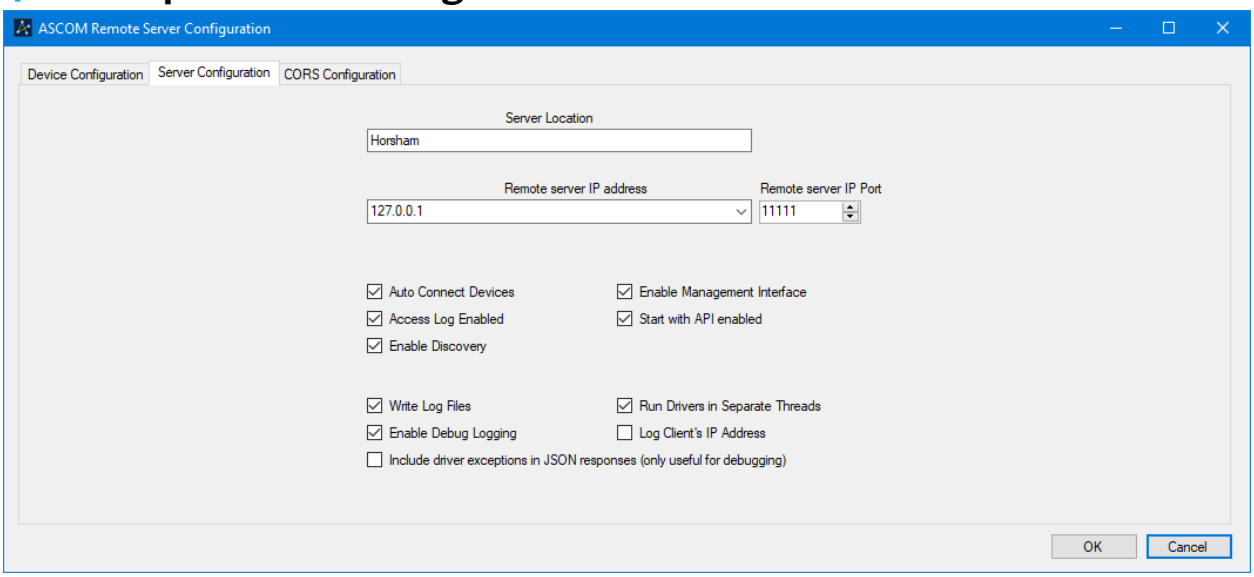


Figure 6 - Remote server configuration dialogue

4.2.1 IP Address and Port

The Remote Server IP Address and Remote Server Port dropdowns enable you to select the IP address and port number on which the server will listen. The IP address list will be pre-populated with all the available network addresses on the host PC plus "localhost".

If you have more than one network adapter on the PC you can also use the "+" or "*" addresses in the dropdown list to listen on all configured network ports.

4.2.2 Discovery

The Remote Server supports the Alpaca Discovery Protocol and listens on the network interfaces configured in the "Remote server IP address" field. The response contains the port number on which the Remote Server URI listener is configured to listen together with a unique ID (a GUID) that identifies this Remote Server instance.

4.2.3 Server Location and Management Interface

The management interface conforms to the Alpaca Management API standard as documented here: Alpaca Management API Specification

It returns information on the remote server as a whole, including the "Server Location" field and a list of the devices configured on the Device Configuration tab. The location field can contain descriptive text such as a physical location or a PC or VM machine name.

The management interface is enabled or disabled through the "Enable management interface" check box.

4.2.4 Remote Server Startup

Whether the Remote Server starts with devices unloaded or loaded and with the listening URI enabled or disabled can be controlled from the "Auto connect devices" and "Start with API enabled" checkboxes.

4.2.5 Drivers in Separate Threads

The "Run Drivers in Separate Threads" checkbox chooses between:

- **Enabled**: Runs each driver in its own thread with exclusive use of a Windows event loop. (Default)
- **Disabled**: Runs all drivers on the Remote Server's UI thread sharing a common Windows event loop.

Running drivers in their own threads is the preferred mode of operation because it provides greater isolation of driver issues from other drivers and from the Remote Server itself. There are currently no known downsides to this approach; the "run all on the main thread" option, however, is provided as a fall back in case of issues arising when using separate threads.

4.3 Setup - CORS Configuration

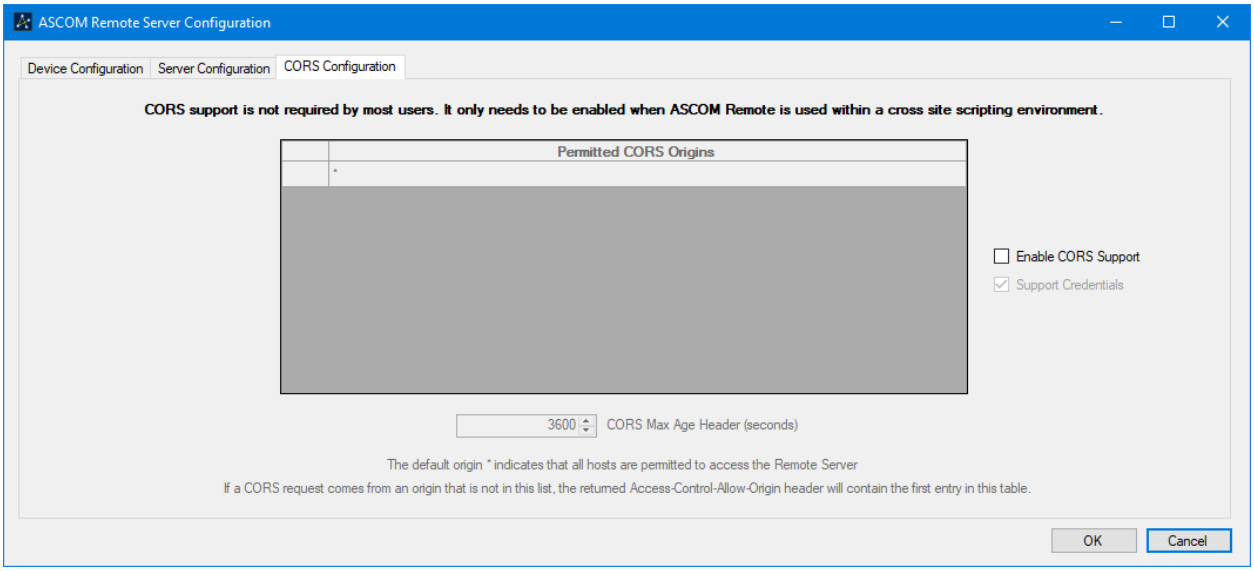


Figure 7 - CORS configuration dialogue

Most people will not need CORS support, it is only required when the ASCOM Remote Server is incorporated as part of a web application that is accessed through a browser.

The default CORS configuration, when enabled, uses a permitted origin of "*", which permits access from all hosts.

The CORS implementation flow diagram is shown over page in Figure 8 - CORS flow diagram.

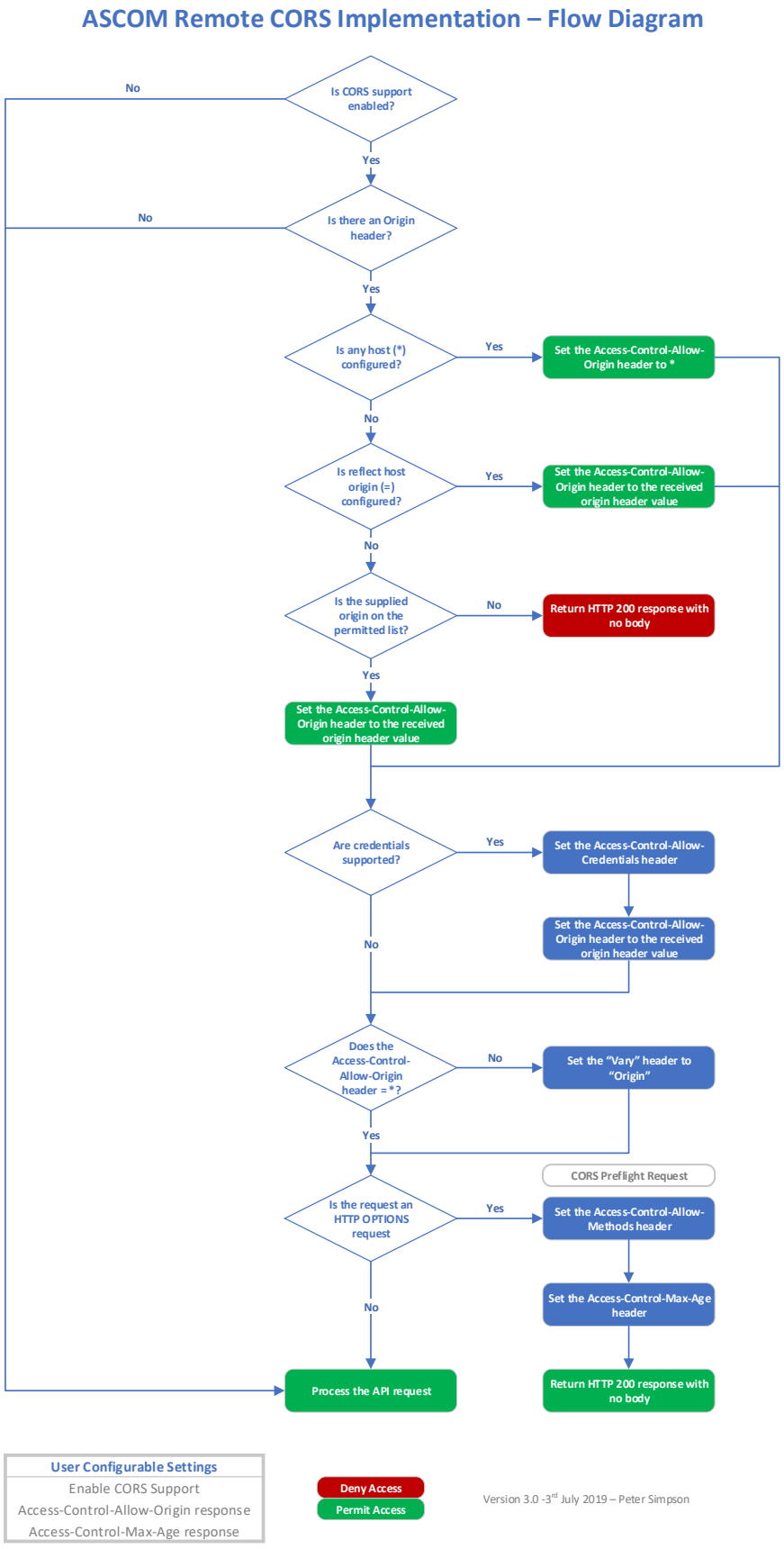


Figure 8 - CORS flow diagram

4.4 Using Camera.ImageArray Base64 Handoff Mode

The ASCOM Remote camera client has built-in support for the base64 handoff mode, which can be enabled through the client setup dialogue.

This section will be of interest to developers creating their own applications e.g. in Python who want to take advantage of the speed increase that the base64 handoff mode provides.

To request use of the base64 handoff mode, if available, the client should add this HTTP header to the HTTP GET /api/v1/camera/x/imagearray request:

`base64handoff = true`

The Remote Server will then request the image from the device and, after it is available, will return a small JSON response similar to Figure 9 - Small base64 handoff JSON response. The JSON response will have the same HTTP header:

`base64handoff = true`

indicating that it supports the base64 handoff mechanic. If the header is absent, the JSON response should be interpreted as the large JSON response containing the image array data that is described in the Alpaca API Specification.

```
{
"Type": 2,
"Rank": 2,
"Dimension0Length": 4,
"Dimension1Length": 4,
"Dimension2Length": 0,
"ClientTransactionID": 0,
"ServerTransactionID": 182,
"ErrorNumber": 0,
"ErrorMessage": ""
}
```

Figure 9 - Small base64 handoff JSON response

The Type, Rank, ID and Error fields are as specified for the Alpaca ImageArray response

The base64string is obtained by an HTTP GET the endpoint:

`/api/v1/camera/x/imagearraybase64`

where x is the camera device number as used in the original GET to the device's imagearray endpoint.

Once received, it is the client's responsibility to decode the base64 string into an array whose dimensions are given in the small JSON response DimensionXLength fields.