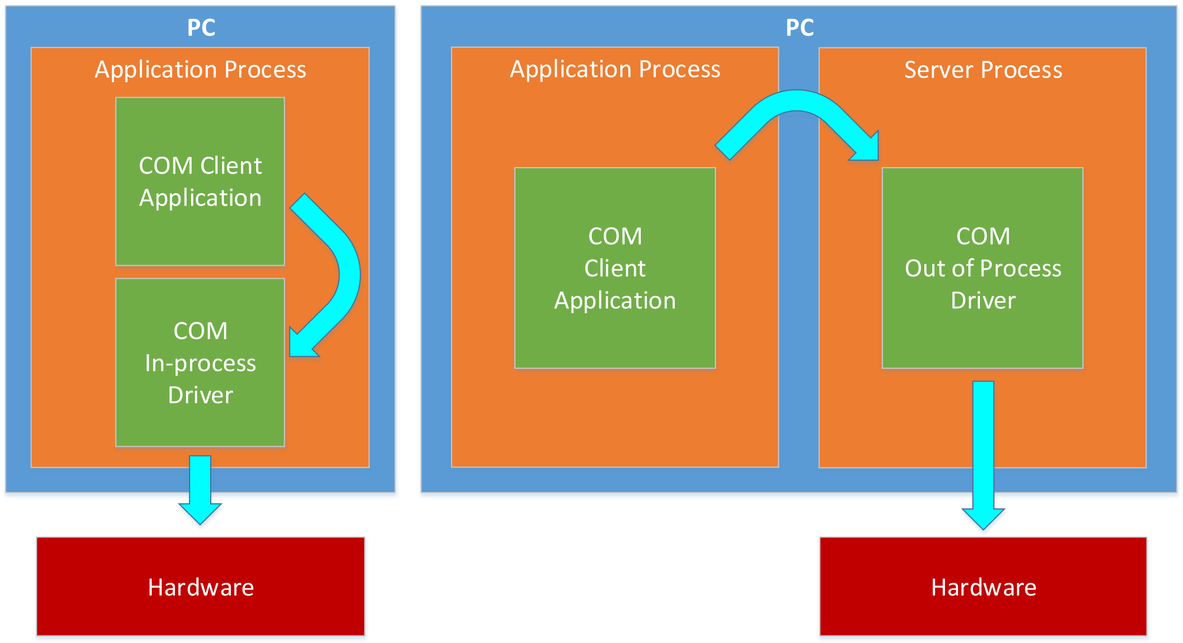
ASCOM Web – Peter Simpson

1 Context

ASCOM's prime USP is the collection of interface definitions that comprise the ASCOM standard. These provide actionable methods on a standard device model, which masks the uniqueness and individuality inherent in real world device control protocols.

Today, ASCOM is limited to Microsoft Windows technology because ASCOM Clients and Drivers communicate with each other through a protocol called COM (Component Object Model). This protocol enables interoperability between clients and drivers written if different development languages but both client and driver must reside on the same PC.

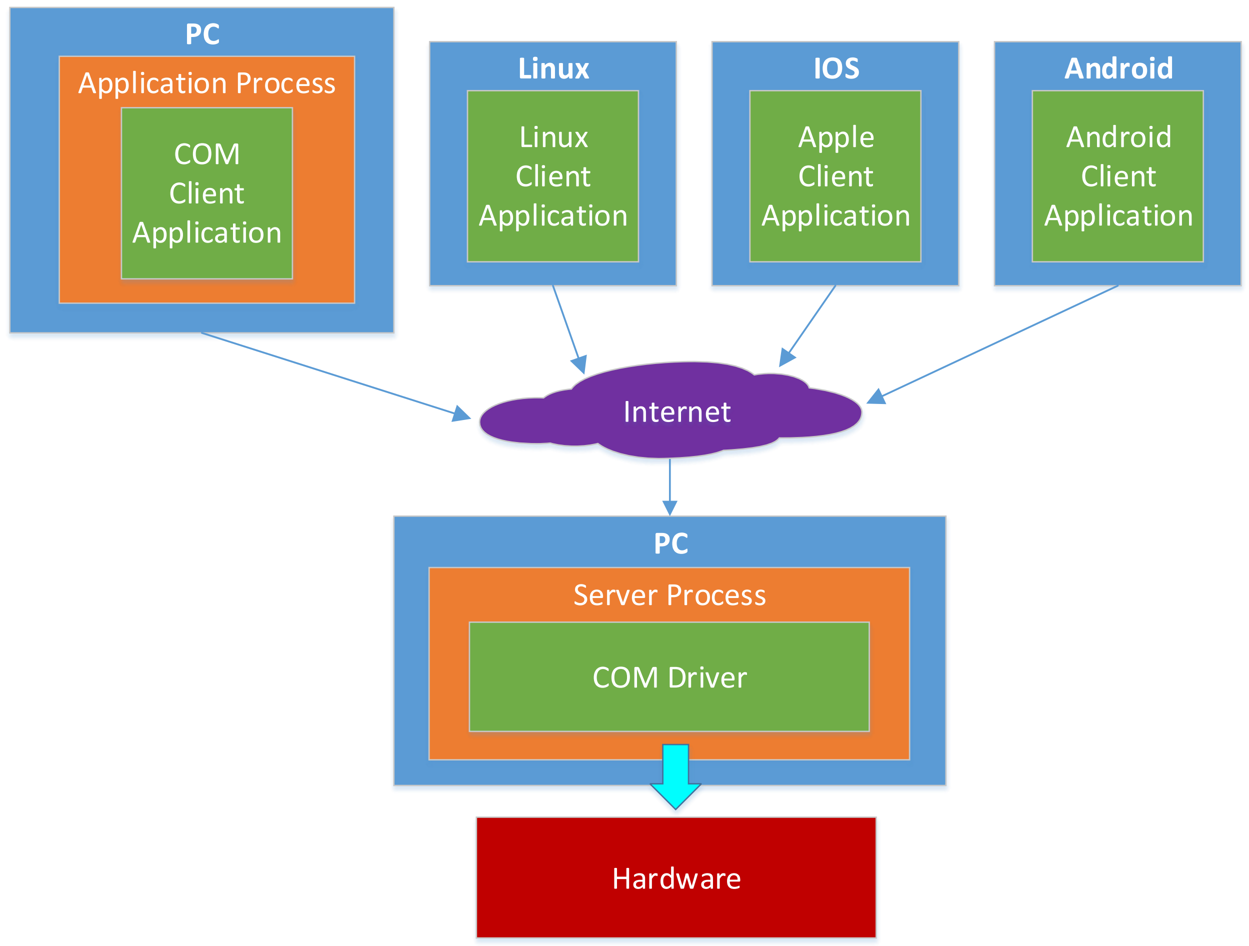
COM servers can be of two kinds: "in process" and "out of process", the key difference is whether the COM server (the ASCOM driver) runs within the calling application's process or outside it in an independent process of its own as shown below.



Microsoft recognized the "same computer" limitation, which applies to both kinds of COM server, and developed DCOM, which proved complex to implement and which, by comparison with current IP based interoperability protocols, did not achieve large scale take up.

In today's heterogeneous world, ASCOM's uni-plaform technology looks increasingly outdated and, to maintain long term relevance, ASCOM needs to support interoperability between clients and drivers running on disparate operating systems such as IOS, Linux, Android etc. as well as Windows.

The following diagram shows this conceptually.



2 High level solution architecture

Realising this conceptual model in practice requires:

Requirement	Approach to realisation
A platform neutral data representation of the ASCOM device APIs	**ASCOM restful interface specification**
A common communications protocol supported by many client devices	**http protocol**
A means to encode the data so that it can be reliably transported by the communications protocol and be easily converted into meaningful data structures within the client.	**JSON data encoding**
A means to protect the data whilst in transit	**https (http over Transport Layer Security or its predecessor, Secure Sockets Layer)**
A means to control access to the remote devices	**http basic authentication**

Two additional ASCOM components are also required:

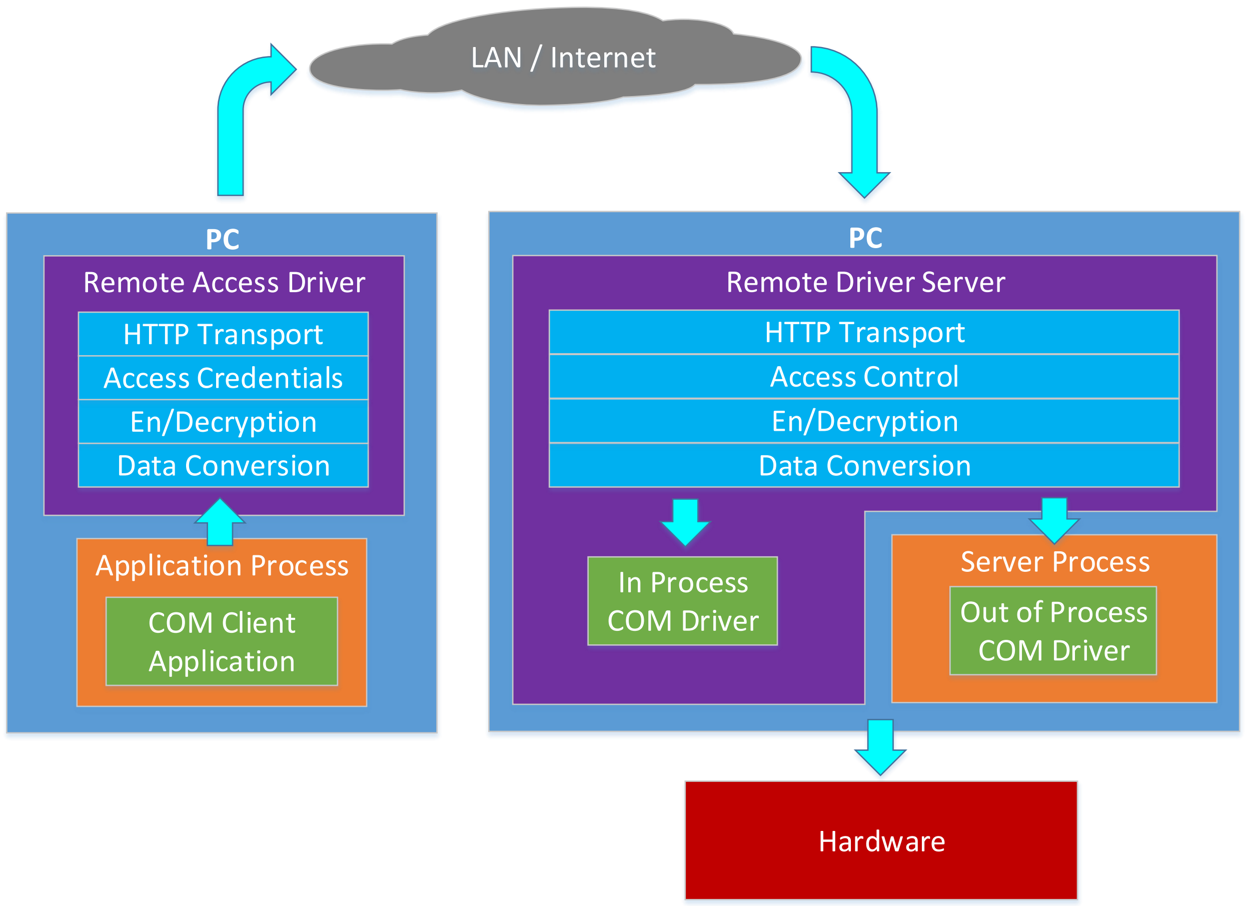
2.1 Remote Access Driver

This appears to client applications as a driver of the required device type e.g. Telescope. This component translates the clients COM requests into the ASCOM REST API standard, handles authentication, encryption and transport of the command to the remote host that houses the remote driver.

The remote access driver can be configured through its Setup Dialogue with required access credentials and the device server's URL or IP address etc.

2.2 Remote Driver Server

This component must be started manually (or through a startup script) and will expose an IP end point able to parse incoming ASCOM Rest API requests and pass these on to the configured drivers, either in process or out of process servers.

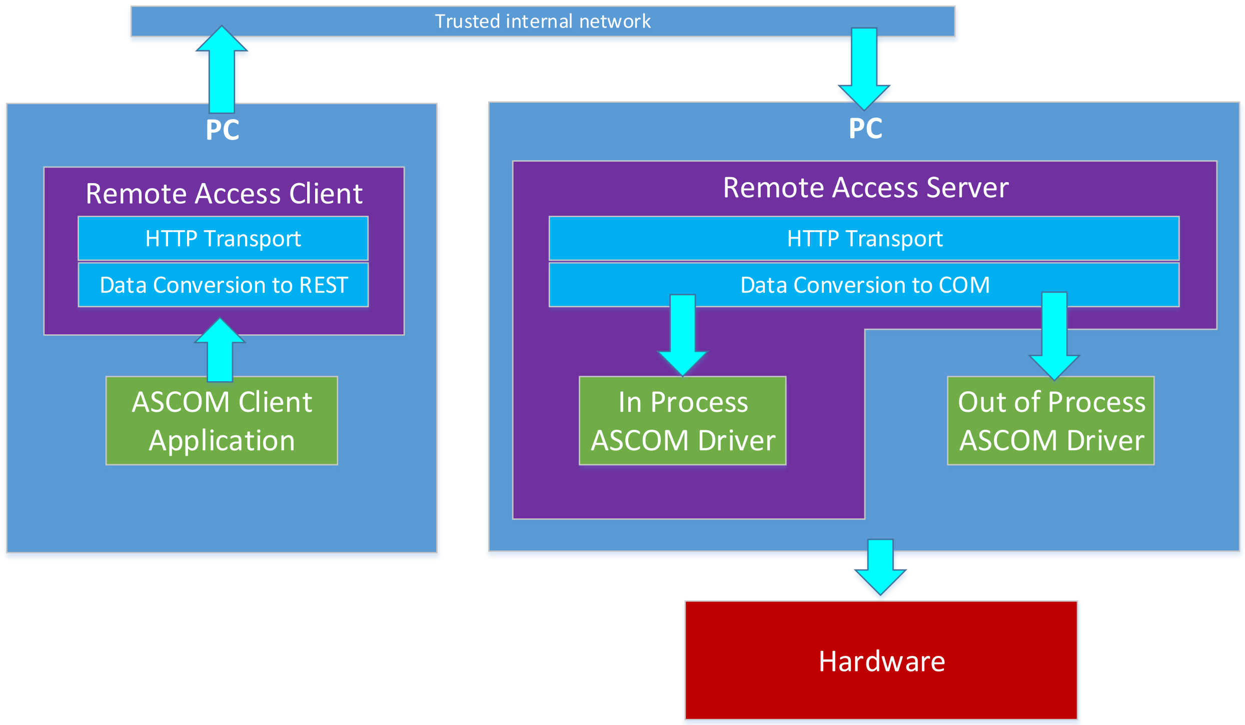


3 Security and the Real World

The remote access driver is able to initiate HTTP or HTTPS connections and also supports redirection to a TLS / SSL HTTPS connection. It can also provide a username and password to support basic authentication.

3.1 Internal Network

This diagram shows a physical realisation of an internal use case, where data does not need to travel over the Internet:



HTTP must be used internally on the trusted network because the remote server cannot terminate SSL/TLS connections.

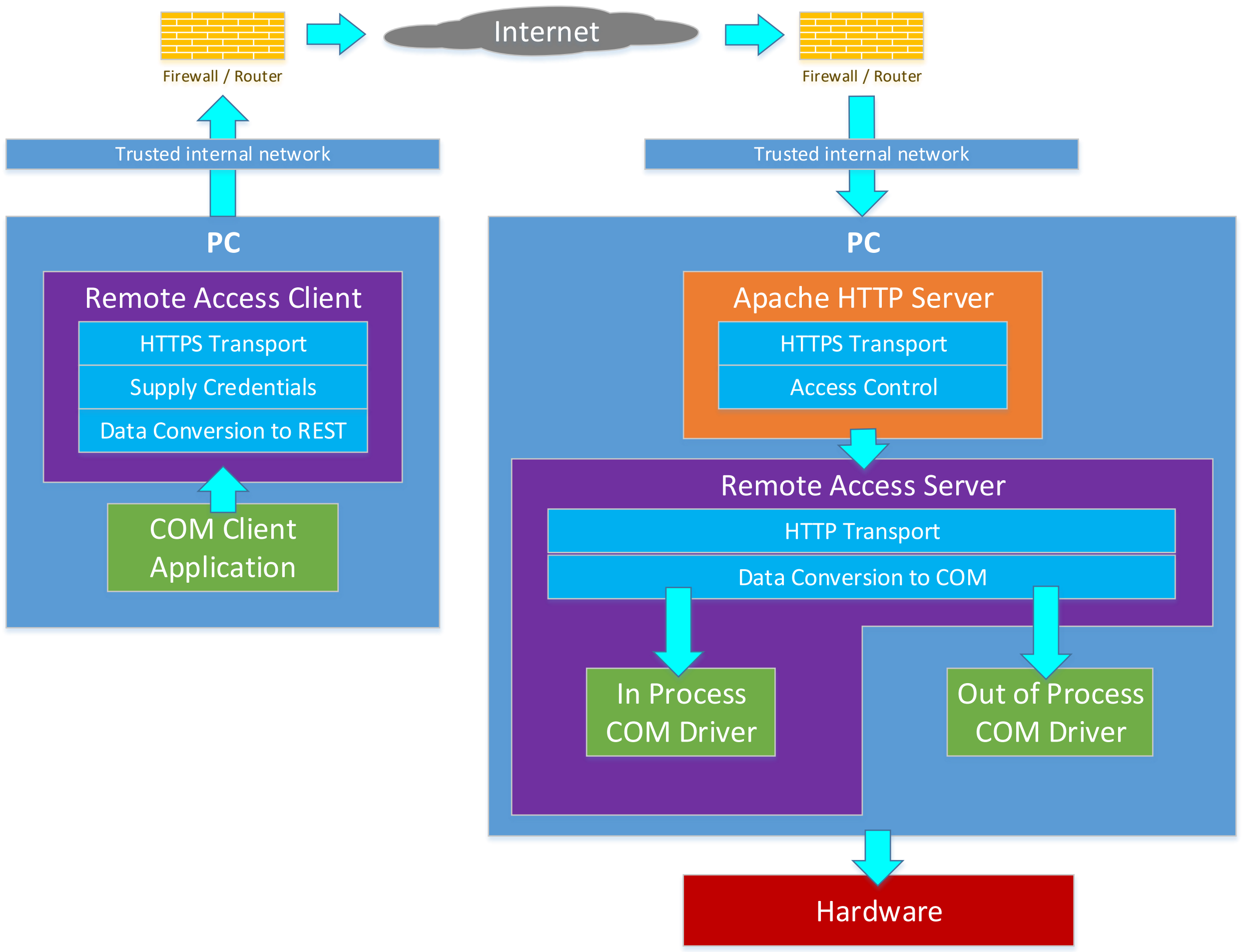
3.2 External Network (Windows Client)

The Remote Server supports HTTP connections but does not support HTTPS connections owing to the complexity of options in this area. An HTTP connection can be set up over the internet just as in the case of the internal network but of course there is no protection for data in transit and also no authentication of the client by the server, so anyone could access the controlled device.

HTTPS is recommended as good practice for communication over the Internet and, if required, the recommended approach is to use a web server such as Apache to handle the secure session termination and to proxy the request to an internal server instance running on a local trusted network.

The Apache instance will also support client authentication as well as handling the complex, ever evolving, SSL/TLS algorithms.

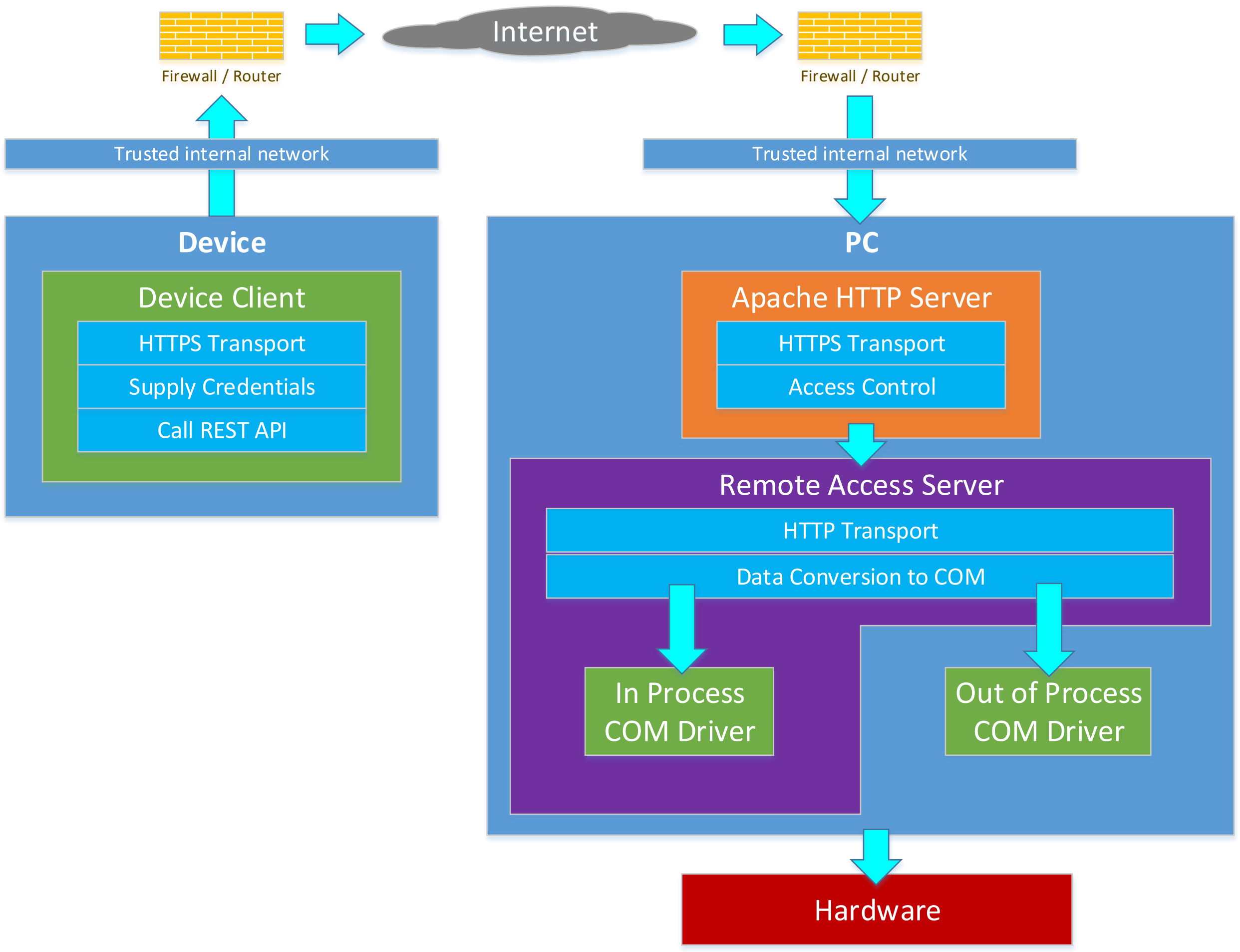
This diagram shows a configuration where remote devices are accessed over the Internet:



3.3 External Network (Non-Windows Client)

One of the key objectives of this initiative is to enable non-Windows devices to access Windows based ASCOM drivers. This diagram illustrates the approach where the device client needs to:

- Connect to the remote access server
- Supply authentication credentials
- Issue appropriate HTTP GET and PUT commands
- Interpret the resultant JSON responses.



4 API Contract

4.1 API Format

The standard API format (fixed text is black and variable elements are red) will be:

http://host/API/devicetype/VVersionNumber/resource?parameters

e.g.: http://api.peakobservatory.com/API/Telescope/V1/AxisRates?Axis=0

e.g.: http://api.peakobservatory.com/API/Telescope/V1/CanSlew

The remote server URI handler is not case sensitive so the first example above could also be:

http://api.peakobservatory.com/api/telescope/v1/axisrates?axis=0

Clients may optionally supply a client ID number and a transaction ID number to identify themselves and this particular transaction. The transaction ID will be returned in the remote server JSON response along with any output from the driver.

4.2 Http Verbs

GET	Used for all information retrieval where the device state is not changed, e.g. most properties and a few functions such as Telescope. AxisRates(Axis).
PUT	is used for all other commands i.e. those which change the state of the device regardless of whether they are properties or methods e.g. setting Telescope.SideOfPier and Telescope.SlewToCoordinates.

4.3 JSON Responses

The outcome of the command is returned as a JSON encoded class. The following information is returned for every transaction:

Item	Type	Contents
ClientTransactionID	Long	Transaction ID supplied by the client in its request
ServerTransactionID	Long	The server's transaction number. This increments by 1 on each call to the server.
Method	String	The name of the method called by the client
ErrorNumber	Int	If the driver throws an exception, its number appears here, otherwise the value 0 is returned. This will be of value to non .NET clients, in order to determine what has occurred, if they have no use for a .NET exception structure.
ErrorMessage	String	If the driver throws an exception, its message appears here, otherwise an empty string is returned.
DriverException	Exception (.NET)	If the driver throws an exception, it is returned as a .NET exception structure encoded in JSON form.

In addition, the JSON response will include the output from the command (if any). This example is from the Telescope Simulator SupportedActions property:

```
GET /api/v1/Telescope/0/SupportedActions?Client=1&ClientTransaction=6
```

{"Value":["AssemblyVersionNumber","SlewToHA","AvailableTimeInThisPointingState","TimeUntilPointingStateCanChange"],"ClientTransactionID":6,"ServerTransactionID":6,"Method":"SupportedActions","ErrorNumber":0,"ErrorMessage":"","DriverException":null}

This example shows the response for:

```
GET /api/v1/Telescope/0/CanSlewAsync?Client=1&ClientTransaction=20
```

{"Value":true,"ClientTransactionID":20,"ServerTransactionID":168,"Method":"CanSlewAsync","ErrorNumber":0,"ErrorMessage":"","DriverException":null}

4.4 Driver Exception Handling

For Windows clients the full driver exception is captured by the server application, and returned in JSON encoded format to the calling driver, which recreates the original exception and throws it to the client application.

This approach is of little value to a non-Windows client so an integer error number and error message string are also returned that can be used by the client as needed, without having to worry about the complex .NET exception class structure.

This example shows a returned exception for an attempt to set the site elevation to -301, which is an illegal value in the ASCOM specification:

```
PUT /api/v1/Telescope/0/SiteElevation
```
(parameters for PUT verbs are placed in the body (not shown here) and do not appear after the URI as they do for GET verbs)

{"ClientTransactionID":51,"ServerTransactionID":58,"Method":"SiteElevation","ErrorNumber":-2147220479,"ErrorMessage":"SiteElevation set - '-301' is an invalid value. The valid range is: -300 to 10000.","DriverException":{"ClassName":"System.Runtime.InteropServices.COMException","Message":"SiteElevation set - '-301' is an invalid value. The valid range is: -300 to 10000.","Data":null,"InnerException":null,"HelpURL":null,"StackTraceString":"   at System.Dynamic.ComRuntimeHelpers.CheckThrowException(Int32 hresult, ExcepInfo& excepInfo, UInt32 argErr, String message)\r\n   at CallSite.Target(Closure , CallSite , ComObject , Double)\r\n   at System.Dynamic.UpdateDelegates.UpdateAndExecute2[T0,T1,TRet](CallSite site, T0 arg0, T1 arg1)\r\n   at CallSite.Target(Closure , CallSite , Object , Double)\r\n   at System.Dynamic.UpdateDelegates.UpdateAndExecute2[T0,T1,TRet](CallSite site, T0 arg0, T1 arg1)\r\n   at ASCOM.Web.RemoteDeviceServer.ServerForm.WriteDouble(String method, HttpRequest request, HttpResponse response) in C:\\Users\\Peter\\Documents\\Visual Studio Projects\\ASCOM Web\\Remote Device Server\\ServerForm.cs:line 997","RemoteStackTraceString":null,"RemoteStackIndex":0,"ExceptionMethod":"8\nCheckThrowException\nSystem.Dynamic, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a\nSystem.Dynamic.ComRuntimeHelpers\nVoid CheckThrowException(Int32, System.Dynamic.ExcepInfo ByRef, UInt32, System.String)","HResult":-2147220479,"Source":"ASCOM.Simulator.Telescope","WatsonBuckets":null}} Errors and HTTP Status codes

4.5 Server Exception Handling

The HTTP status code will reflect the following statuses:

Code	Interpretation	
200	**OK**	Successful GET or PUT for a synchronous command (even if the driver returned an exception)
400	**Bad request**	At least one of the device type, version, resource or parameters to be supplied by the client are missing or invalid.
500	**Internal Server Error**	Errors occurring in the Remote Driver Server itself

5 Notes

- **OS must be Windows Vista or later**: The new components are written in .NET 4.6 and this means that Windows operating systems must be Vista or later.
- **Device.Connected:** The remote device server can be configured to honour or disregard connection instructions from clients. This enables a device to be maintained in a connected

state even if the client disconnects, useful when there are multiple concurrent clients or when the device is intended as an “always available” service. In the “always on” configuration Windows clients will see Telescope.Connected changing state as they expect, but the real state will not change.

6 Installation and Use

The installer has two install options, one to install the remote device server and the other to install Windows client drivers that can access the server and make it appear as if it the devices are locally connected. The installer will:

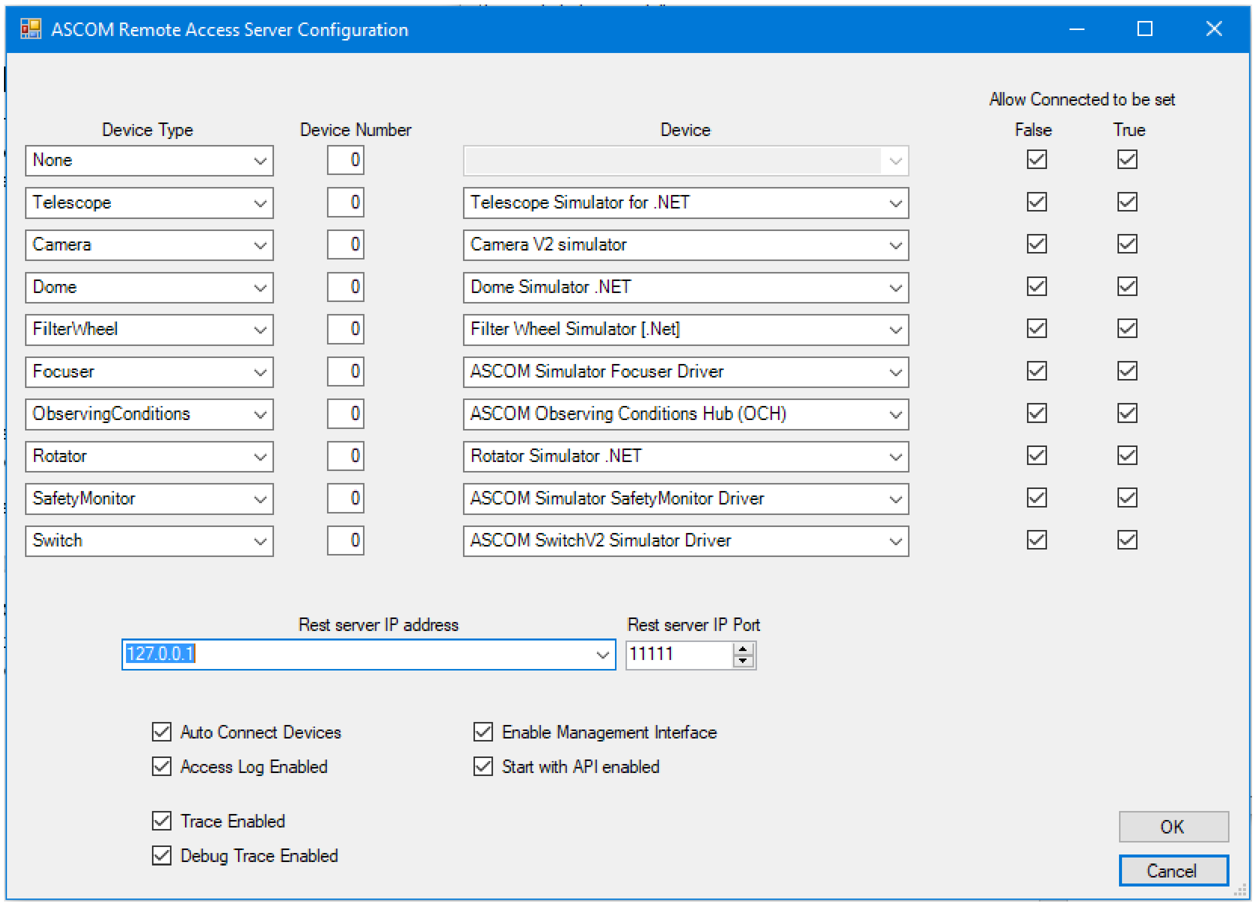
- Install a new device drivers e.g. ASCOM.Web1.Telescope, which appear as ASCOM Web Client Driver 1 in Chooser.
  *** Note: At present only one driver is installed but future versions will install a second driver to allow for two remote devices of the same type.
- Install the remote driver server in Start/All Programs/ASCOM Remote Driver Server.

6.1 Configure the server

There is a “Setup” button the server form that brings up a setup dialogue to allow device types and devices to be selected.

There is a dropdown to enable you to select the IP address and port number on which the server will listen. This should be pre-populated with all the available network addresses on the host PC plus “localhost”.

The Device Number is automatically assigned as device types are selected and **this, along with the Device Type, is what has to be configured into the client** in order to speak with the correct served device.



ASCOM Remote Access Server Configuration

Device Type	Device Number	Device	Allow Connected to be set: False	Allow Connected to be set: True
None	0		✓	✓
Telescope	0	Telescope Simulator for .NET	✓	✓
Camera	0	Camera V2 simulator	✓	✓
Dome	0	Dome Simulator .NET	✓	✓
FilterWheel	0	Filter Wheel Simulator [.Net]	✓	✓
Focuser	0	ASCOM Simulator Focuser Driver	✓	✓
ObservingConditions	0	ASCOM Observing Conditions Hub (OCH)	✓	✓
Rotator	0	Rotator Simulator .NET	✓	✓
SafetyMonitor	0	ASCOM Simulator SafetyMonitor Driver	✓	✓
Switch	0	ASCOM SwitchV2 Simulator Driver	✓	✓

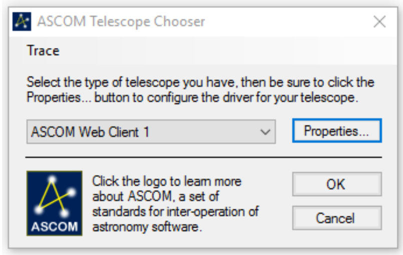
Rest server IP address: 127.0.0.1
Rest server IP Port: 11111

✓ Auto Connect Devices
✓ Access Log Enabled
✓ Enable Management Interface
✓ Start with API enabled
✓ Trace Enabled
✓ Debug Trace Enabled

6.2 Configure the Clients

These are normal ASCOM drivers (local server model) that appear as ASCOM Web Client 1 in each device type and that can be configured through Chooser as normal. There are fields to

- select the connection type (HTTP or HTTPS). *Please note that the remote server itself only supports http)*
- IP address and port name of the remote server. *A host name should work but has not yet been tested. For testing its fine to run the driver and the remote server on the same PC and localhost is the best choice of address for this configuration.*
- Timeouts (in seconds) for quick response commands such as CanXXX properties and slow response commands such as SlewToCoordinates.
- Fields to enter username and password for authentication *(Only required if Apache or some other web server is used to proxy incoming remote device server connections and the web server has been configured to require a password to access one of the URIs. Username and password are not required on the EC2 demo system.)*



ASCOM Telescope Chooser

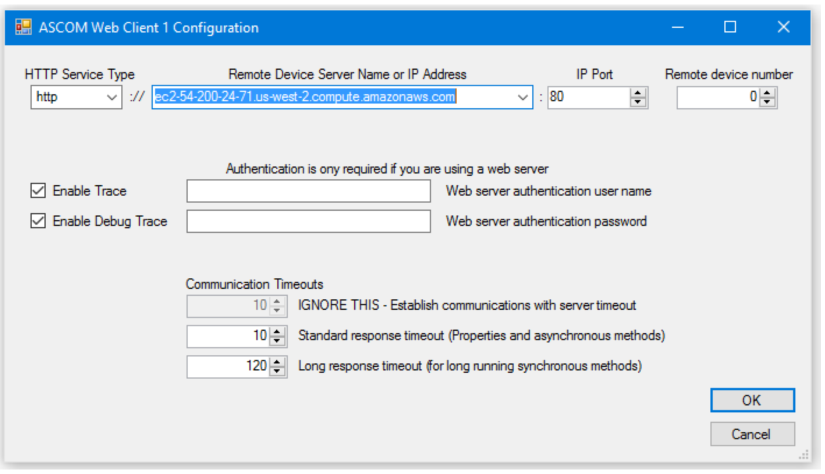
Trace

Select the type of telescope you have, then be sure to click the Properties... button to configure the driver for your telescope.

ASCOM Web Client 1 | Properties...

Click the logo to learn more about ASCOM, a set of standards for inter-operation of astronomy software.

OK | Cancel



ASCOM Web Client 1 Configuration

HTTP Service Type: http :// Remote Device Server Name or IP Address: ec2-54-200-24-71.us-west-2.compute.amazonaws.com : IP Port: 80 Remote device number: 0

Authentication is ony required if you are using a web server

☑ Enable Trace — Web server authentication user name

☑ Enable Debug Trace — Web server authentication password

Communication Timeouts

10 IGNORE THIS - Establish communications with server timeout

10 Standard response timeout (Properties and asynchronous methods)

120 Long response timeout (for long running synchronous methods)

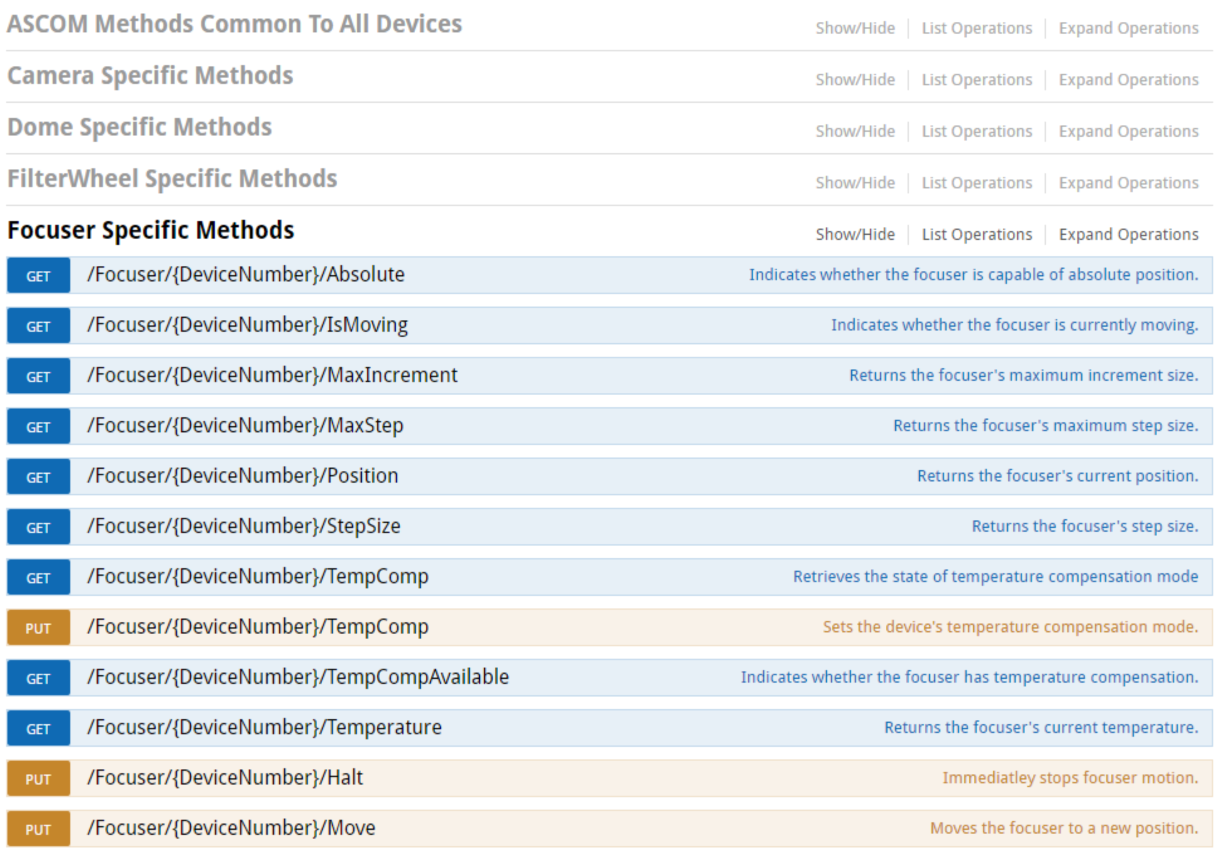
OK | Cancel

6.3 API Documentation

Swagger documentation for all methods is available through this URL:

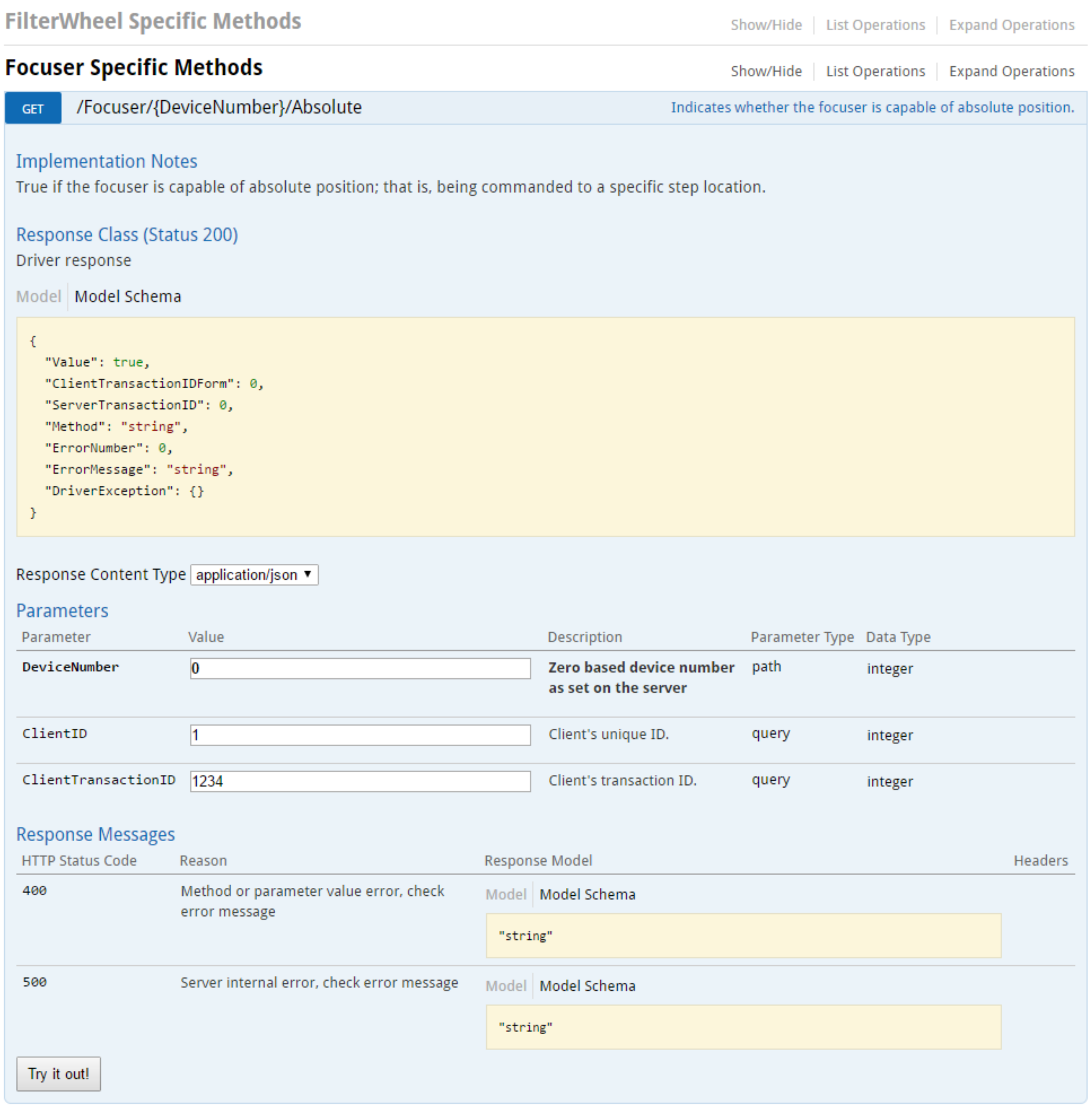
http://www.ascom-standards.org/api

click on one of the grey Show/Hide links to expand one of the sets of methods.



ASCOM Methods Common To All Devices		Show/Hide | List Operations | Expand Operations
Camera Specific Methods		Show/Hide | List Operations | Expand Operations
Dome Specific Methods		Show/Hide | List Operations | Expand Operations
FilterWheel Specific Methods		Show/Hide | List Operations | Expand Operations
Focuser Specific Methods		Show/Hide | List Operations | Expand Operations
GET	/Focuser/{DeviceNumber}/Absolute	Indicates whether the focuser is capable of absolute position.
GET	/Focuser/{DeviceNumber}/IsMoving	Indicates whether the focuser is currently moving.
GET	/Focuser/{DeviceNumber}/MaxIncrement	Returns the focuser's maximum increment size.
GET	/Focuser/{DeviceNumber}/MaxStep	Returns the focuser's maximum step size.
GET	/Focuser/{DeviceNumber}/Position	Returns the focuser's current position.
GET	/Focuser/{DeviceNumber}/StepSize	Returns the focuser's step size.
GET	/Focuser/{DeviceNumber}/TempComp	Retrieves the state of temperature compensation mode
PUT	/Focuser/{DeviceNumber}/TempComp	Sets the device's temperature compensation mode.
GET	/Focuser/{DeviceNumber}/TempCompAvailable	Indicates whether the focuser has temperature compensation.
GET	/Focuser/{DeviceNumber}/Temperature	Returns the focuser's current temperature.
PUT	/Focuser/{DeviceNumber}/Halt	Immediatley stops focuser motion.
PUT	/Focuser/{DeviceNumber}/Move	Moves the focuser to a new position.

Then click on one of the blue Get methods e.g.:


FilterWheel Specific Methods

Show/Hide | List Operations | Expand Operations

Focuser Specific Methods

Show/Hide | List Operations | Expand Operations

/Focuser/{DeviceNumber}/Absolute

Indicates whether the focuser is capable of absolute position.

Implementation Notes

True if the focuser is capable of absolute position; that is, being commanded to a specific step location.

Response Class (Status 200)

Driver response

```
{
  "Value": true,
  "ClientTransactionIDForm": 0,
  "ServerTransactionID": 0,
  "Method": "string",
  "ErrorNumber": 0,
  "ErrorMessage": "string",
  "DriverException": {}
}
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
DeviceNumber	0	**Zero based device number as set on the server**	path	integer
ClientID	1	Client's unique ID.	query	integer
ClientTransactionID	1234	Client's transaction ID.	query	integer

Response Messages

HTTP Status Code	Reason	Response Model	Headers
400	Method or parameter value error, check error message	Model Model Schema "string"	
500	Server internal error, check error message	Model Model Schema "string"	

The "Try it out!" button is not functional at present because the content needs to be hosted from a server running Windows and the ASCOM Remote Device Server.