ASCOM REMOTE

Concept, Configuration and Use

Abstract

ASCOM Remote breaks the limitation that ASCOM clients and drivers must be installed on the same PC and enables non-Windows clients to use Windows ASCOM drivers. This is achieved by middleware that transparently communicates through TCP/IP over internal networks or the Internet.

Peter Simpson
peter@peterandjill.co.uk

Contents

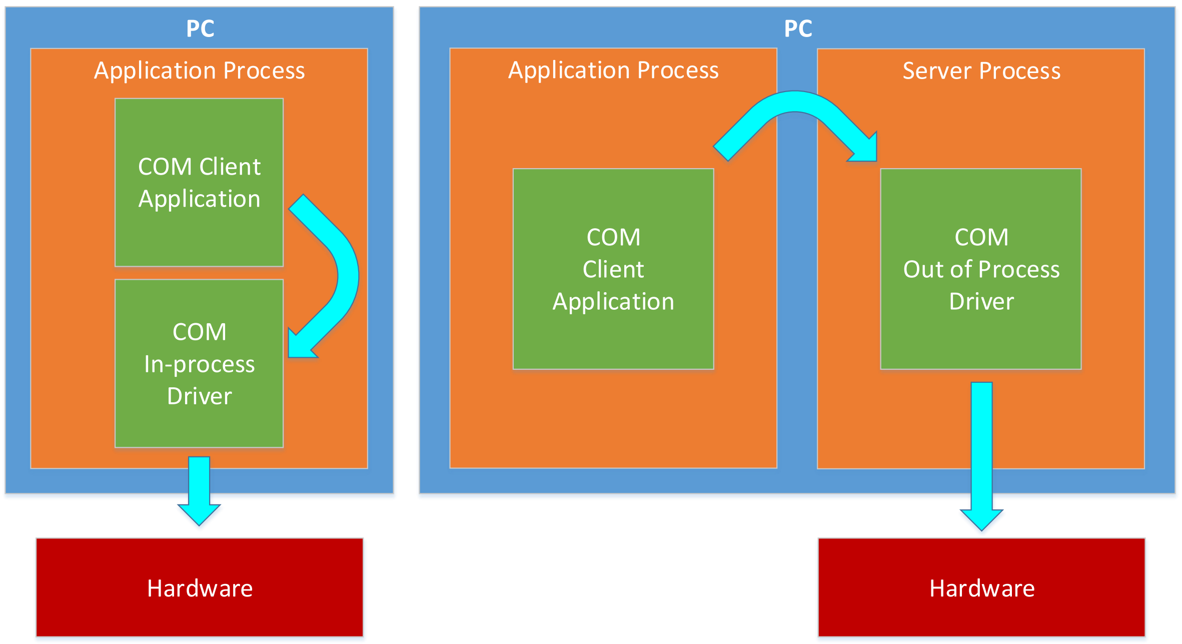
ASCOM Remote

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.

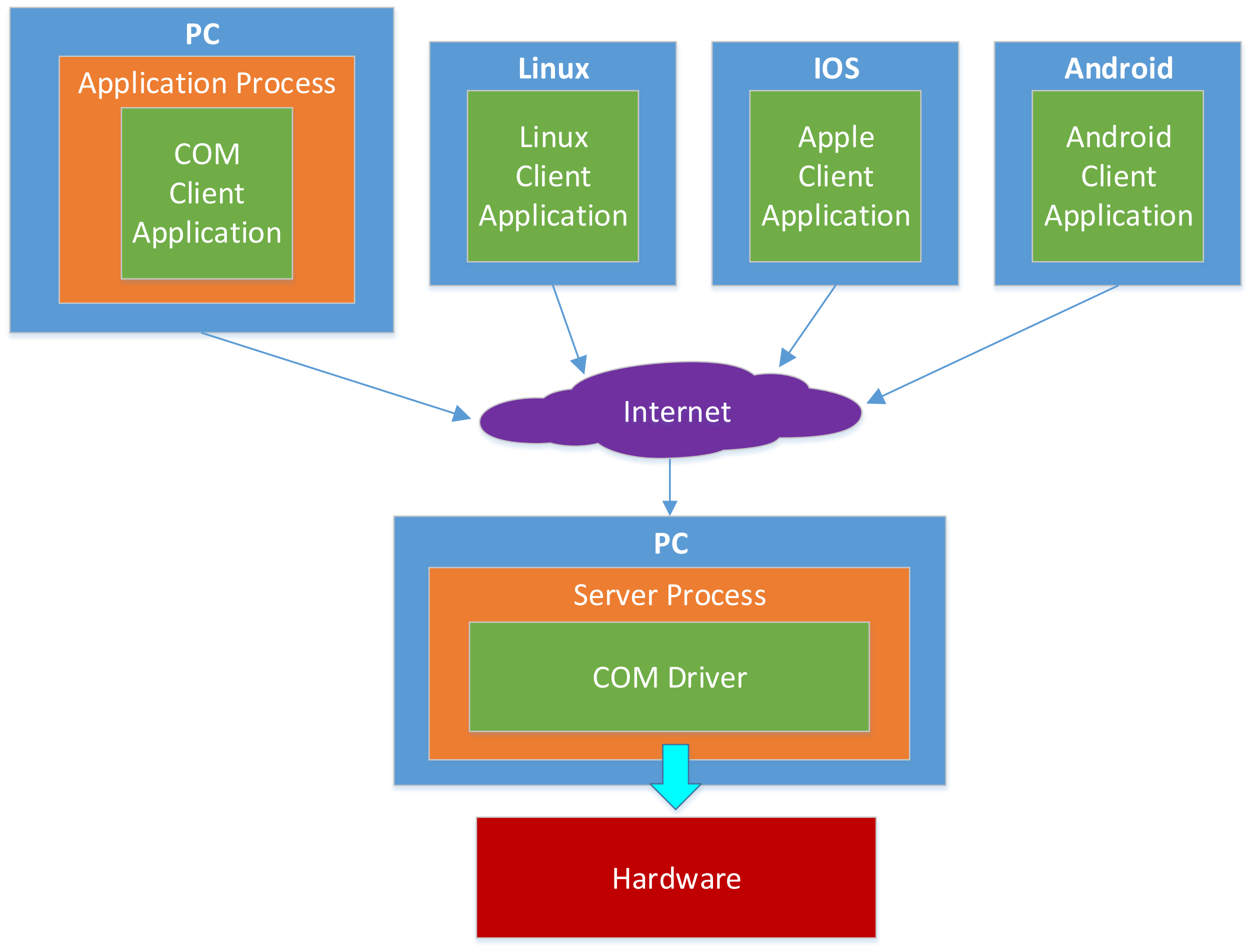
COM servers can be of two kinds: "in process" and "out of process", the key difference is whether the COM server (the ASCOM driver) runs within the calling application's process or outside it in an independent process of its own as shown below.



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 over TCP/IP**
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:

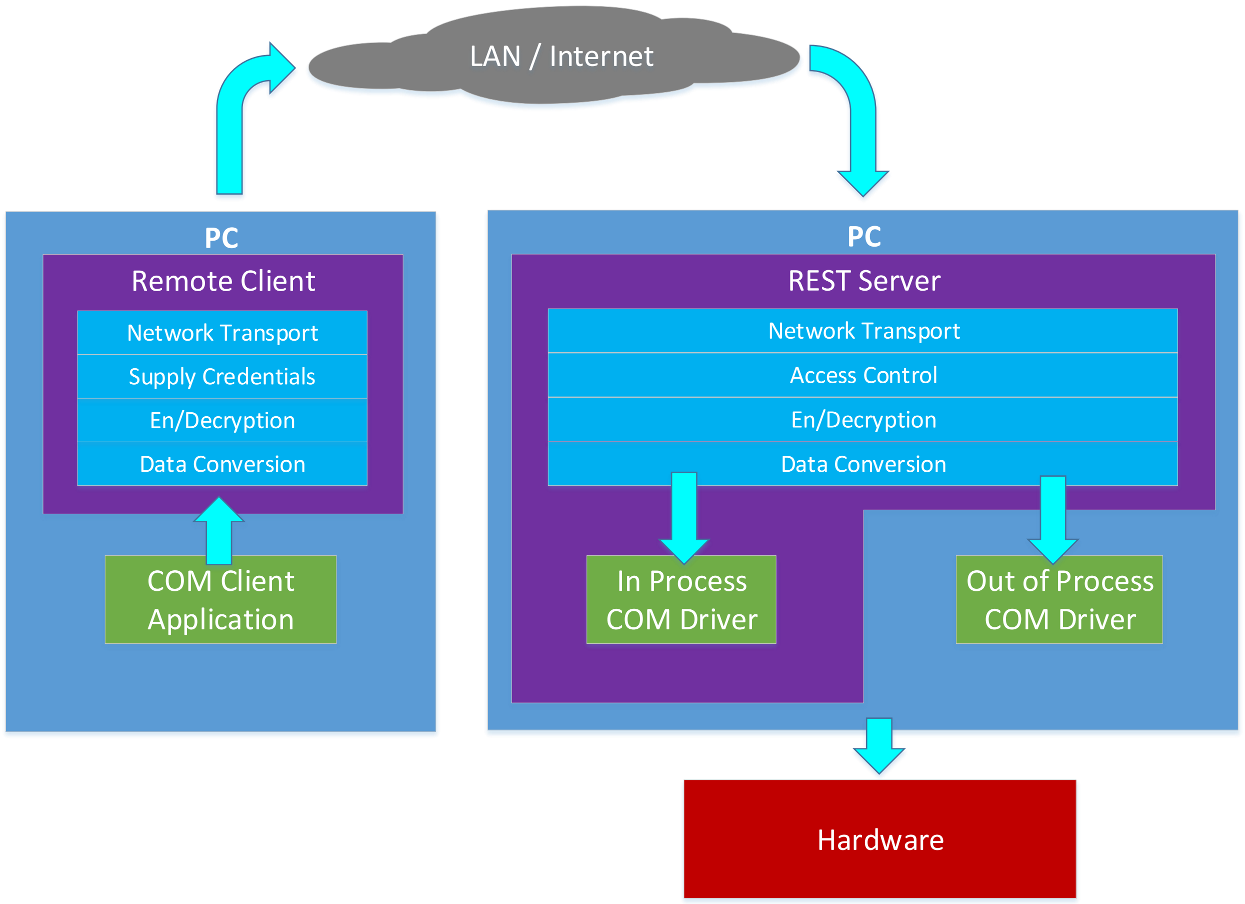
2.1 Remote Client

This appears to client applications as a driver of the required device type e.g. Telescope. This component translates the client's 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 client can be configured through its driver Setup Dialogue with required access credentials and the device server's URL or IP address etc.

2.2 REST Server

This component exposes an IP end point and translates incoming ASCOM REST API requests back to COM requests before passing them to the configured ASCOM drivers.

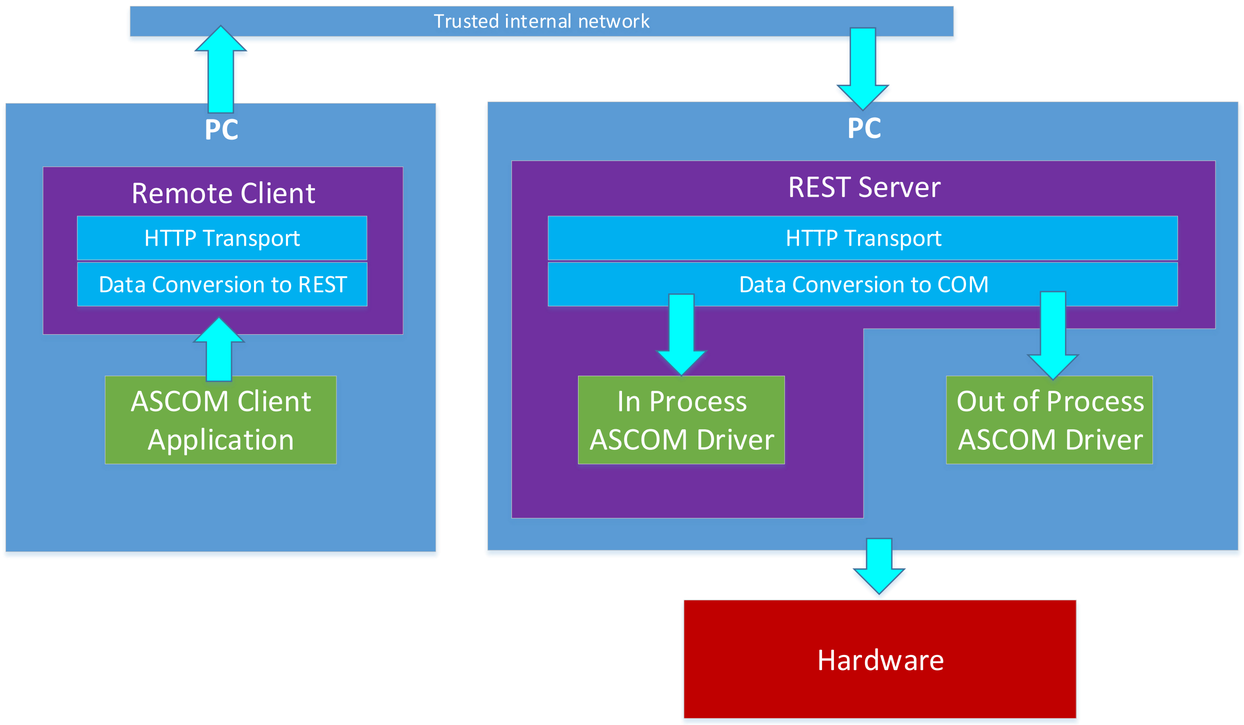


3 Security and the Real World

The Remote Client can initiate HTTP or HTTPS connections and supports redirection from HTTP to a TLS / SSL HTTPS connection. If required, it can supply a username and password to support basic HTTP 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 REST Server cannot terminate SSL/TLS connections.

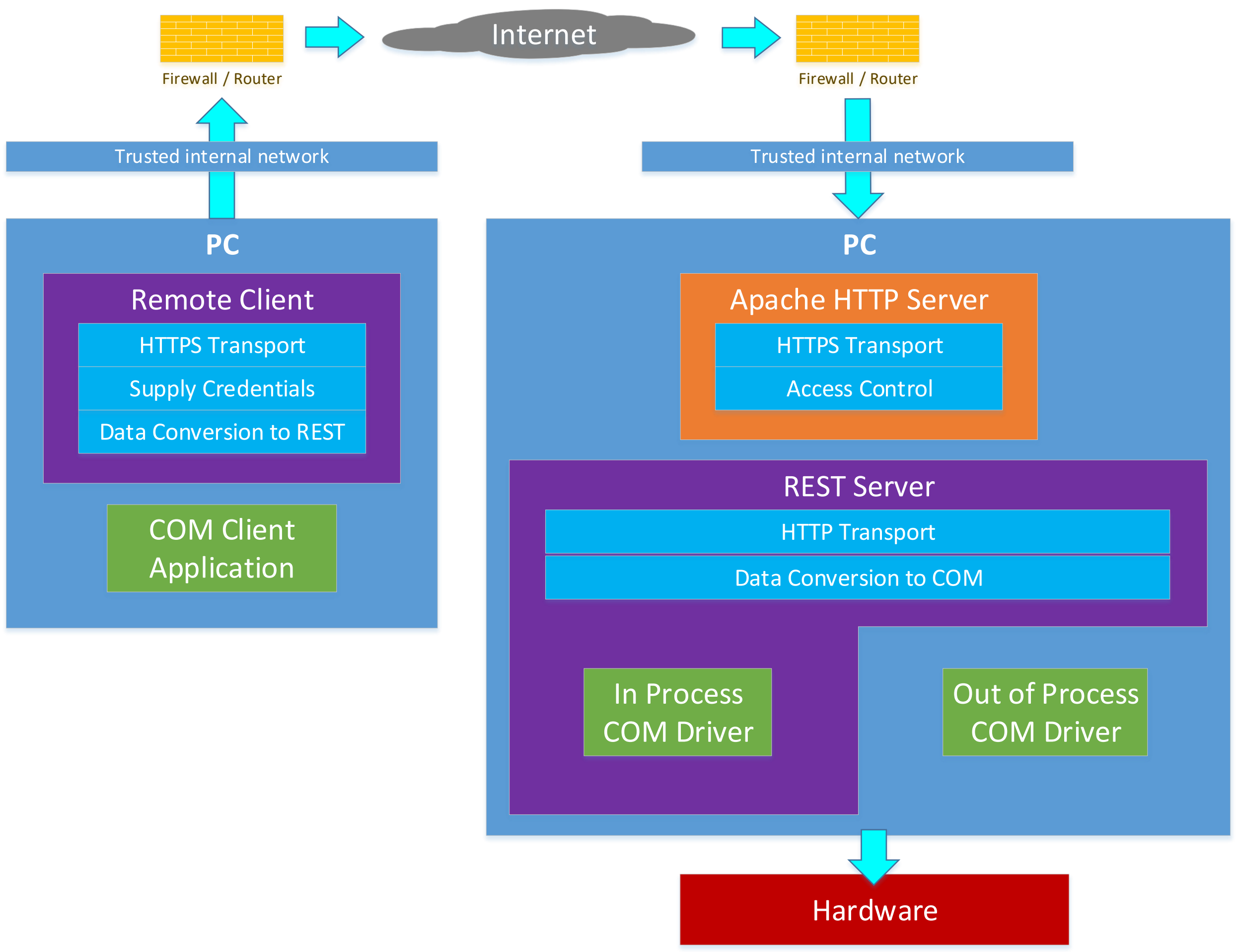
3.2 External Network (Windows Client)

The REST 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.

Use of HTTPS is considered good practice for communication over the Internet and, if required, the recommended approach is to use a web server ahead of the REST Server, such as Apache, to handle secure session termination and proxy requests to the REST 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.

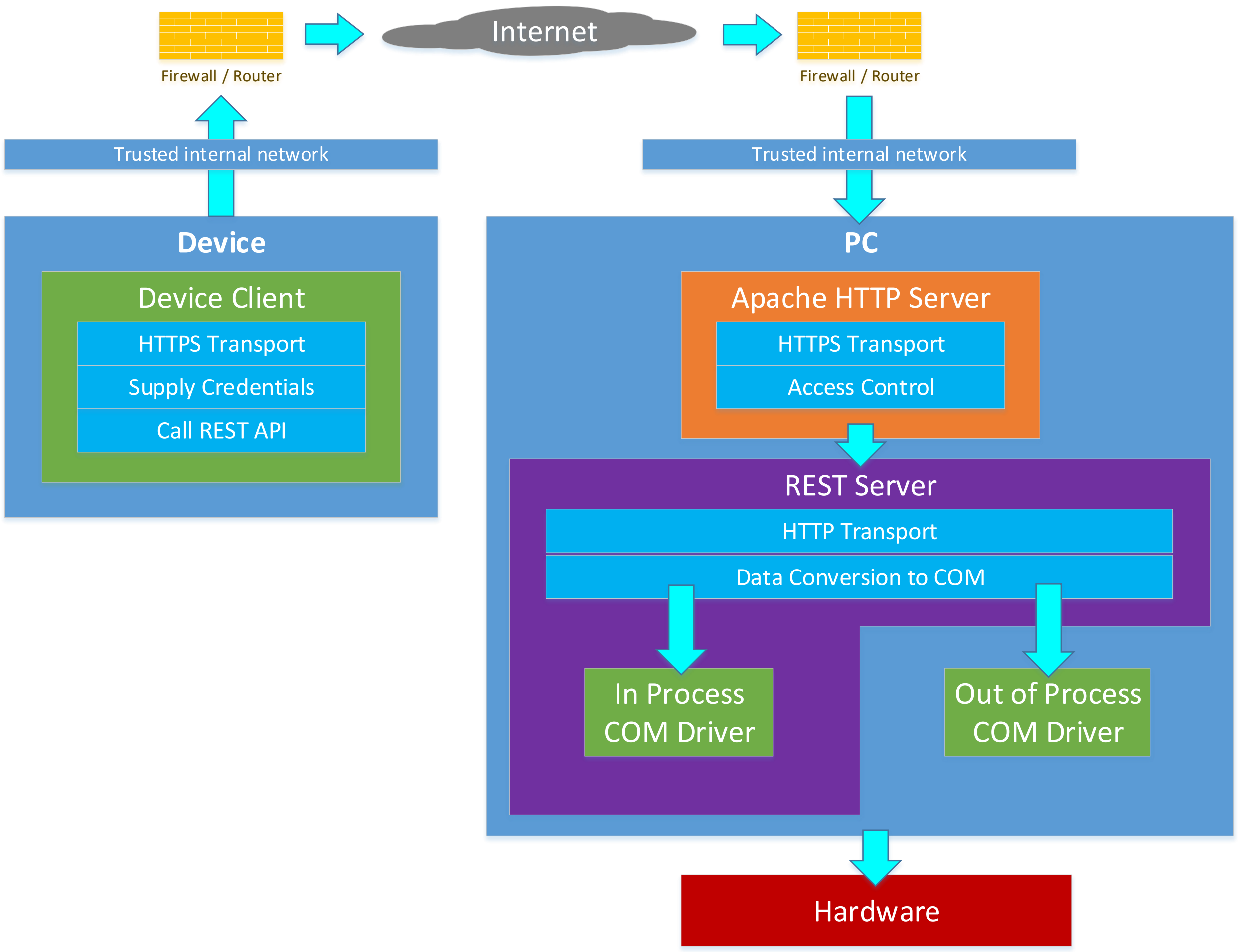
This diagram shows a configuration where remote devices are accessed over the Internet:



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 sent as:

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 since 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) in the "Value" parameter. 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 driver exception is captured by the REST Server and returned in JSON encoded format to the Remote Client, 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, which can be used by the client as needed, without having to use the complex .NET exception class structure.

The example below 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 the PUT verb are placed in the body (not shown here) and do not appear after the URI as they do for the GET verb)

The response contains the error number and message:

```
{"ClientTransactionID":51,"ServerTransactionID":58,"Method":"SiteElevation",
"ErrorNumber":-2147220479,
"ErrorMessage":"SiteElevation set - '-301' is an invalid value. The valid range is: -300 to 10000."
```

As well as the exception itself:

{"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 REST Server Exception Handling

The returned HTTP status code reflects the REST Server's status as follows:

Code	Interpretation	
200	**OK**	The REST Server successfully processed the GET or PUT command (even if the driver returned an exception)
400	**Bad request**	The REST Server cannot process this API request because at least one of the device type, version, resource or parameter values, supplied by the client, are missing or invalid.
500	**Internal Server Error**	An error occurred in the Remote Driver Server itself, which prevented successful processing of the request.

5 Installation and Use

5.1 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.

5.2 Installation

The installer provides options to install either the Remote Clients, the REST 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.

5.3 Configure the Remote Server

Run the server, which is located in Start/ ASCOM Remote, and use the “Setup” button to start the setup dialogue enabling served device types and devices to be selected.

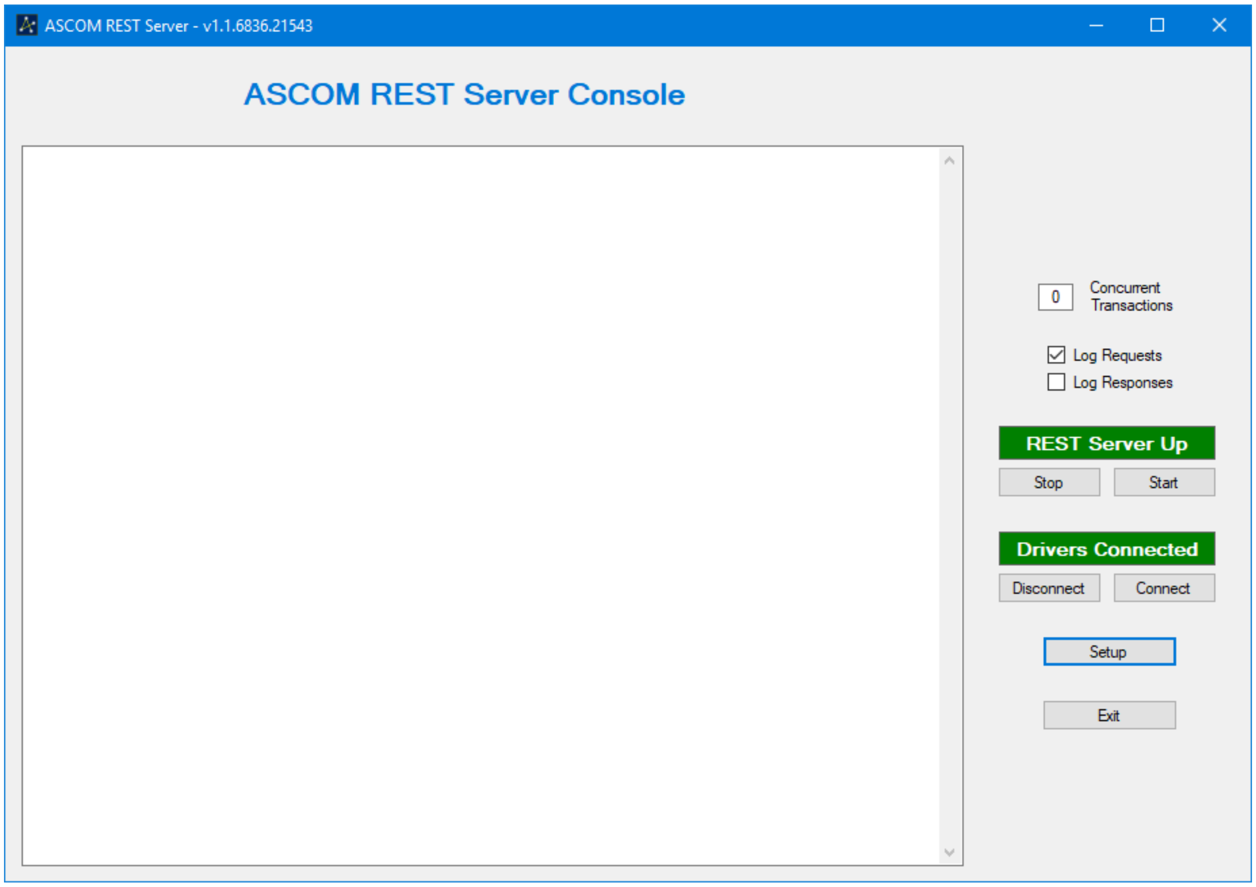
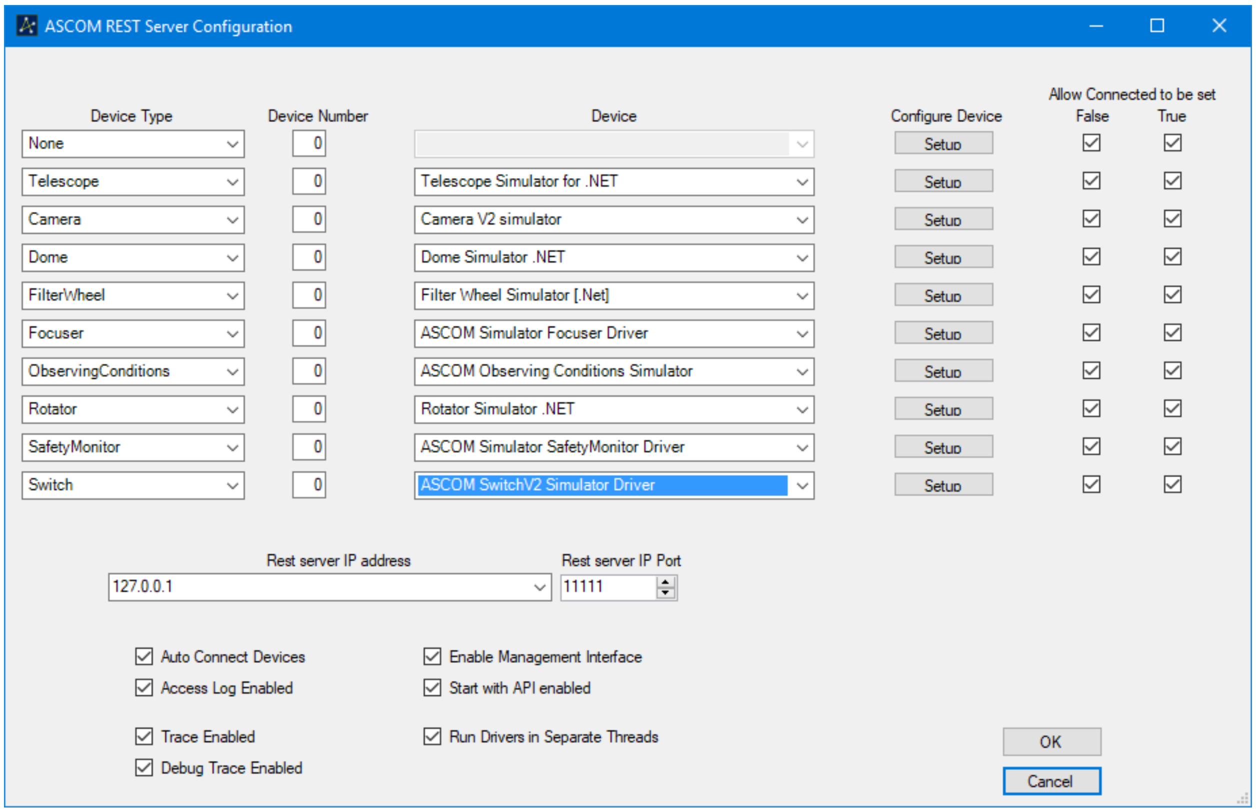


Figure 1 - Remote server console



ASCOM REST Server Configuration

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

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

- ✓ Auto Connect Devices
- ✓ Access Log Enabled
- ✓ Trace Enabled
- ✓ Debug Trace Enabled
- ✓ Enable Management Interface
- ✓ Start with API enabled
- ✓ Run Drivers in Separate Threads

Figure 2 - Remote server configuration form

5.3.1 IP Address

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”.

5.3.2 Device Selection

To configure 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” drop-downs 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.

5.3.3 Connected Settings

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, which can be useful when there are multiple concurrent clients or when the device is intended as an “always available” service.

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.

5.3.4 Management Interface

The management interface provides some capabilities to manage the server remotely:

- Return the ASCOM Profile on the remote PC
- Return the Remote Server device configuration
- Return the instantaneous number of concurrent transactions that the server is handling
- Unload and reload the served drivers

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

5.3.5 Drivers in Separate Threads

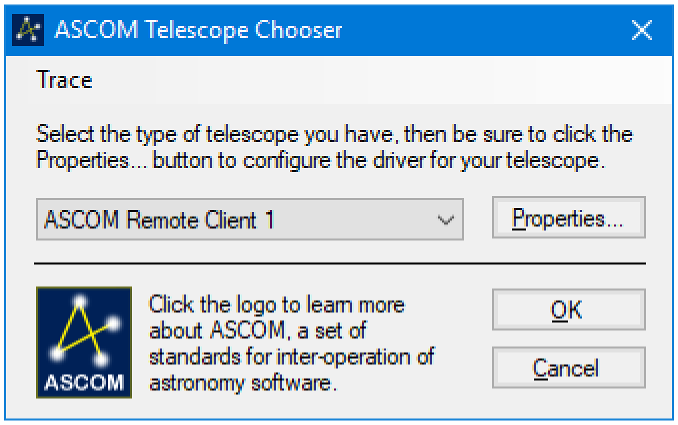
The "Run Drivers in Separate Threads" checkbox chooses between:

- Running each driver in its own separate thread with its own independent Windows event loop. (Default)
- Running all drivers on the Remote Server's main 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.

5.4 Remote Clients

These appear as normal ASCOM drivers named ASCOM Remote Client 1, ASCOM Remote Client 2 etc. that 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.



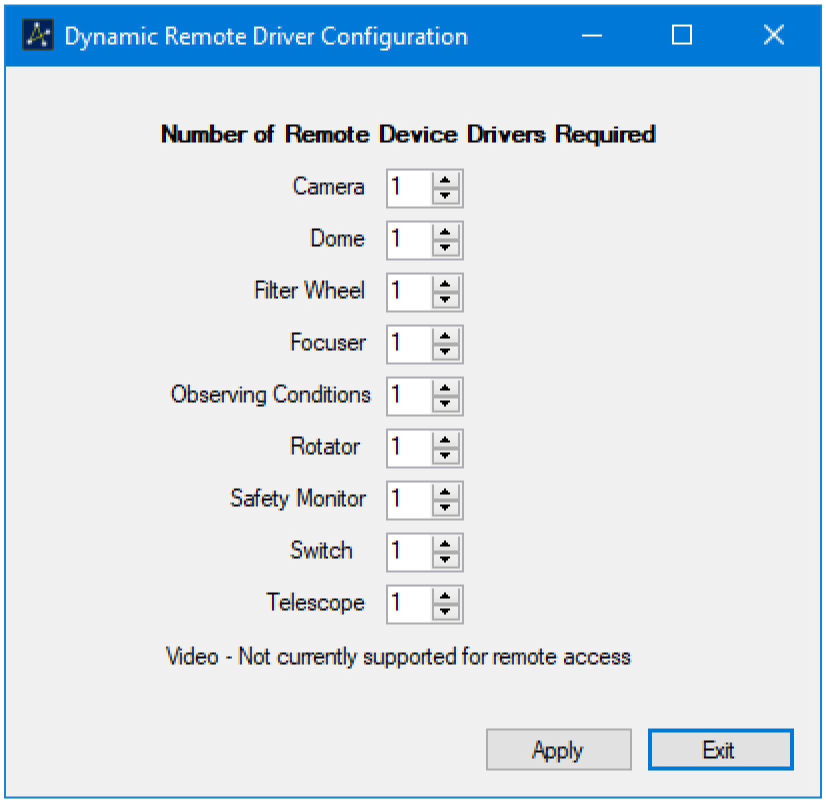
Select the type of telescope you have, then be sure to click the Properties... button to configure the driver for your telescope.

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

Figure 3 - Telescope Chooser showing a remote client

5.4.1 Configuring the number of remote clients

Initially there will be one remote client in 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.



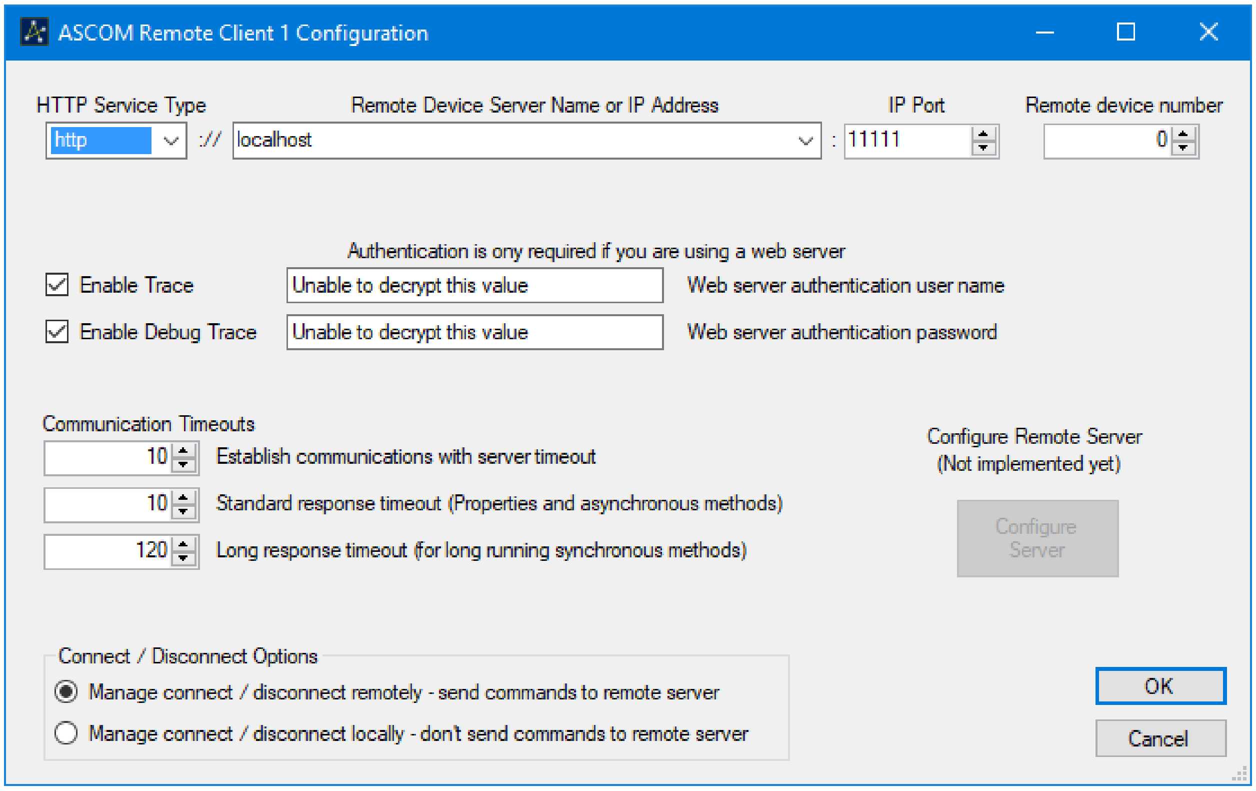
Dynamic Remote Driver Configuration

Number of Remote Device Drivers Required

Camera	1
Dome	1
Filter Wheel	1
Focuser	1
Observing Conditions	1
Rotator	1
Safety Monitor	1
Switch	1
Telescope	1

Video - Not currently supported for remote access

Figure 4 - Utility to configure the number of drivers of each device type



ASCOM Remote Client 1 Configuration

HTTP Service Type: http :// Remote Device Server Name or IP Address: localhost : IP Port: 11111 Remote device number: 0

Authentication is ony required if you are using a web server

☑ Enable Trace — Unable to decrypt this value — Web server authentication user name

☑ Enable Debug Trace — Unable to decrypt this value — Web server authentication password

Communication Timeouts

10 Establish communications with server timeout

10 Standard response timeout (Properties and asynchronous methods)

120 Long response timeout (for long running synchronous methods)

Configure Remote Server
(Not implemented yet)

Connect / Disconnect Options

● Manage connect / disconnect remotely - send commands to remote server

○ Manage connect / disconnect locally - don't send commands to remote server

Figure 5 - Remote client configuration form

5.4.2 Communication with the Remote Server

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

Notes

- The remote server itself only supports HTTP, see Section 3 Security and the Real World
- For testing its fine to run the driver and the remote server on the same PC and to use localhost as the IP address for both clients and server

There are two communication timeouts, one for standard response commands such as CanXXX properties and one 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.

5.4.3 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 required 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.

6 API Documentation

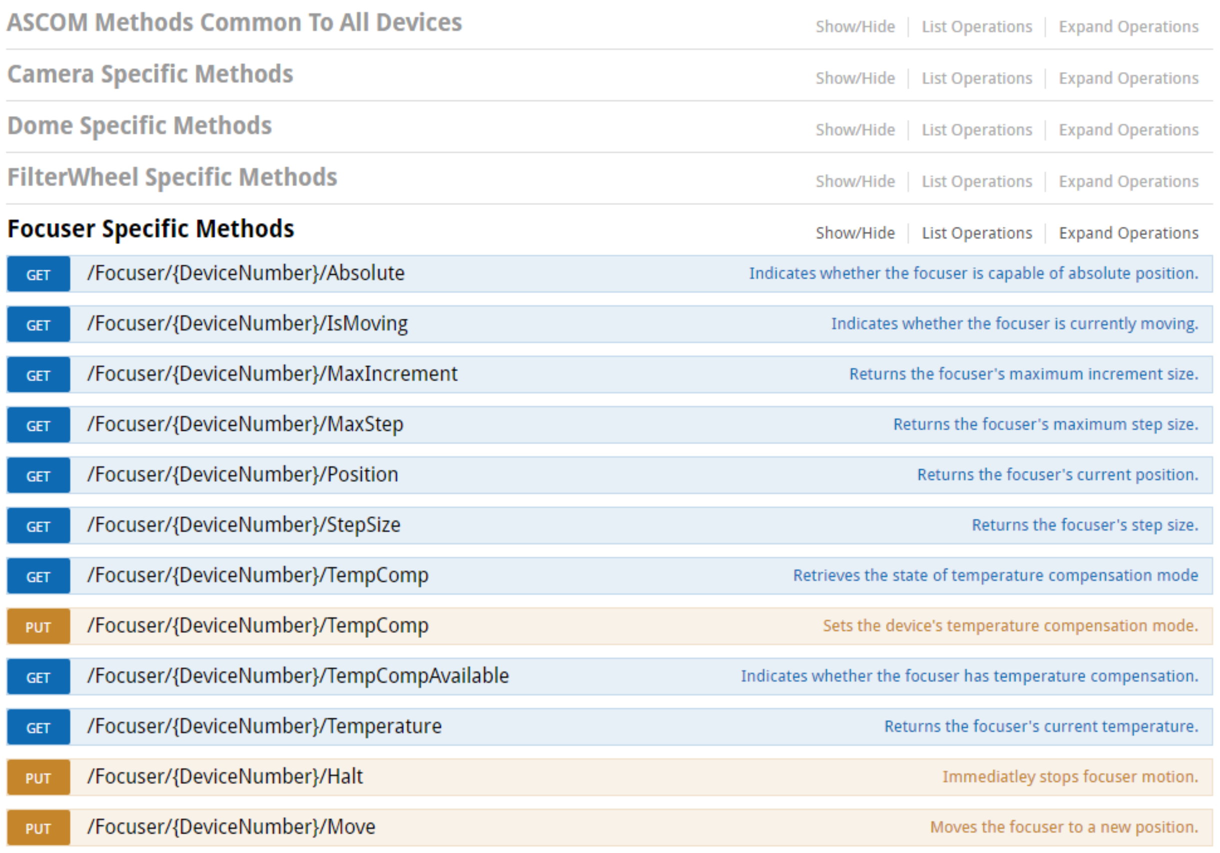
The Remote Server APIs are documented using the Swagger toolset and are available through URLs on the ASCOM Standards web site. The device access API is documented here:

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

and the server management API is documented here:

https://www.ascom-standards.org/api/server.html

To start exploring go to the device access or management API URL and click a grey Show/Hide link to expand one of the sets of methods and then click the blue GET or orange PUT methods for detailed information on that API call.



ASCOM Methods Common To All Devices Show/Hide | List Operations | Expand Operations

Camera Specific Methods Show/Hide | List Operations | Expand Operations

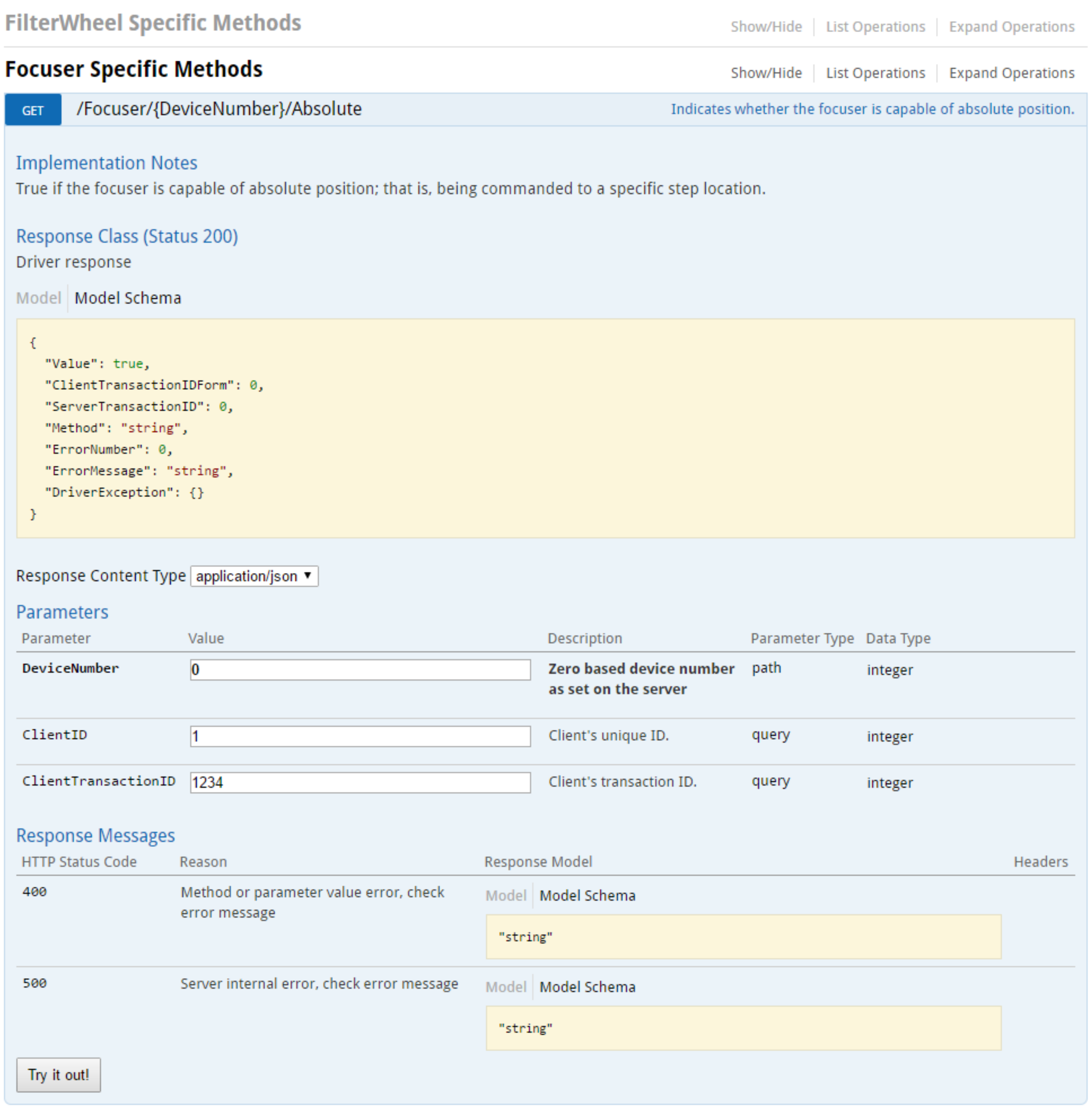
Dome Specific Methods Show/Hide | List Operations | Expand Operations

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.

Figure 6 - Swagger Documentation - Summary level


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.

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

Model | Model Schema

```
{
  "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"	

Figure 7 - Swagger Documentation - Method detail

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.