pyastroimageview Documentation

Release 0.0.1

Michael Fulbright

Nov 02, 2020

CONTENTS:

CHAPTER

ONE

INTRODUCTION

1.1 What is Pyastroimageview?

Pyastroimageview is an image acquisition program that allows taking images with a supported astronomy hardware (via ASCOM or INDI) including:

- cameras
- mounts
- focusers
- filter wheels

Pyastroimageview can take an image sequence and in addition pyastroimageview will control PHD2 to cause dither events while taking an image sequence.

Pyastroimageview also has an JSON-RPC-like API that allows it to act as a hub for hardware like cameras which do not always support multiple clients.

CHAPTER

TWO

PYASTROIMAGEVIEW RPC API DOCUMENTATION

Pyastroimageview supports a JSON-RPC-like API for clients to take images, etc.

2.1 Format

The format of an RPC request is:

```
{ 'method' : <method>,
  'id' : <request_id>,
  'params' : <params>
}
where:
  <method> :  String name of API method being requested
  <request_id> : Request id number - should be incremented for each request and not␣
↪reused
  <params> : JSON dictionary of parameters required for request
```

The format of a response is:

```
{ 'jsonrpc' : '2.0',
  'id' : <request_id>,
  'result' : <result>
}
where:
  <request_id> : Request id number from the request which produced this result
  <result> : JSON dictionary of result of request
```

If an error occured instead the response will be:

```
{ 'jsonrpc' : '2.0',
  'id' : <request_id>,
  'error' : { 'code' : <error_code>, 'message' : <error_message }
}
where:
  <request_id> : Request id number from the request which produced this result
  <error_code> : Error code for exception
  <error_message> : Error message for exception
```

2.2 Supported Methods

2.2.1 Camera

get_camera_info: Accepts:

```
Nothing.
```

Returns:

```
{ 'result' : {
                'binning' : <binning>,
                'framesize' : <framesize>,
                'roi' : <roi>
              }
}
where:
   <binning> : (Integer) Camera binning
   <framesize> : (List) (Camera width, Camera Height)
   <roi> : (List) (Leftmost X, Uppermost Y, Width, Height)
```

get_camera_x_pixelsize: Accepts:

```
Nothing.
```

Returns:

```
{ 'result' : {
                'get_camera_x_pixelsize' : <size>
              }
}
where:
   <size> : (Float) Camera pixel X size in microns
```

get_camera_y_pixelsize: Accepts:

```
Nothing.
```

Returns:

```
{ 'result' : {
                'get_camera_y_pixelsize' : <size>
              }
}
where:
   <size> : (Float) Camera pixel Y size in microns
```

get_camera_max_binning: Accepts:

```
Nothing.
```

Returns:

```
{ 'result' : {
                'get_camera_max_binning' : <maxbin>
              }
```

(continues on next page)

(continued from previous page)

```
}
where:
   <maxbin> : (Integer) Maximum binning supported
```

get_camera_egain: Accepts:

```
Nothing.
```

Returns:

```
{ 'result' : {
               'get_camera_egain' : <egain>
             }
}
where:
   <egain> : (Float) Camera gain in electrons/ADU
```

get_camera_gain: Accepts:

```
Nothing.
```

Returns:

```
{ 'result' : {
               'get_camera_gain' : <gain>
             }
}
where:
   <gain> : (Float) Camera gain
```

get_current_temperature: Accepts:

```
Nothing.
```

Returns:

```
{ 'result' : {
               'current_temperature' : <curtemp>
             }
}
where:
   <curtemp> : (Float) Current camera temperature
```

get_target_temperature: Accepts:

```
Nothing.
```

Returns:

```
{ 'result' : {
               'get_target_temperature' : <targtemp>
             }
}
where:
   <targtemp> : (Float) Current camera target temperature
```

set_target_temperature: Accepts:

```
{
  'target_temperature' : <temperature>
}
where:
    <temperature> : (Float) Target cooler temperature
```

Returns:

```
{
  'complete' : true,
}
```

get_cooler_state: Accepts:

```
Nothing.
```

Returns:

```
{ 'result' : {
               'get_cooler_state' : <state>
             }
}
where:
   <state> : (Boolean) Whether cooler is on (True) or off (False)
```

get_cooler_power: Accepts:

```
Nothing.
```

Returns:

```
{ 'result' : {
               'get_cooler_power' : <power>
             }
}
where:
   <power> : (Float) Current camera cooler power level
```

set_cooler_state: Accepts:

```
{
  'cooler_state' : <state>
}
where:
   <state> : (Boolean) Whether cooler is on (true) or off (false)
```

Returns:

```
{
  'complete' : true,
}
```

take_image: Accepts:

```
{
  'exposure' : <exposure>,
  'binning' : <binning>,
```

(continues on next page)

(continued from previous page)

```
  'roi' : <roi>,
  'frametype' : <frametype'
}
where:
   <exposure> : (Float) Exposure time in seconds
   <binning> : (Integer) Camera binning
   <roi> : (List) ROI for exposure - (Leftmost X, Uppermost Y, Width, Height)
   <frametype> : (String) 'Light', 'Dark', 'Bias', or 'Flat **NOTE** Only 'Light'␣
↪supported!
```

Returns:

```
{
  'complete' : true,
}
```

save_image: Accepts:

```
{
  'filename' : <filename>
}
where:
   <filename> : (String) Output filename including path if required
```

Returns:

```
{
  'complete' : true,
}
```

2.2.2 Focuser

focuser_get_absolute_position: Accepts:

```
Nothing.
```

Returns:

```
{ 'result' : {
               'focuser_get_absolute_position' : <position>
             }
}
where:
   <position> : (Integer) Focuser absolute position
```

focuser_get_max_absolute_position: Accepts:

```
Nothing.
```

Returns:

```
{ 'result' : {
               'focuser_get_max_absolute_position' : <max_position>
             }
}
```

(continues on next page)

(continued from previous page)

```
where:
   <max_position> : (Integer) Maximum allowed absolute position
```

focuser_get_current_temperature: Accepts:

```
Nothing.
```

Returns:

```
{ 'result' : {
               'focuser_get_current_temperature' : <curtemp>
             }
}
where:
   <curtemp> : (Float) Current focuser sensor temperature
```

focuser_is_moving: Accepts:

```
Nothing.
```

Returns:

```
{ 'result' : {
               'focuser_is_moving' : <moving>
             }
}
where:
   <moving> : (Boolean) Whether focuser is currently moving or not
```

focuser_stop:

Accepts: :: Nothing.

Returns:

```
{ 'result' : {
               'stop' : <result>
             }
}
where:
   <result> : (Boolean) Success or not
```

focuser_move_absolute_position: Accepts:

```
{
  'absolute_position' : <position>
}
where:
    <position> : (Integer) Absolute position to move focuser to
```

Returns:

```
{
  'complete' : true,
}
```

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search