FPC

DevicePlugin

Programmer's reference

ImagingControl Camera AQ

Revision 2017-1-23

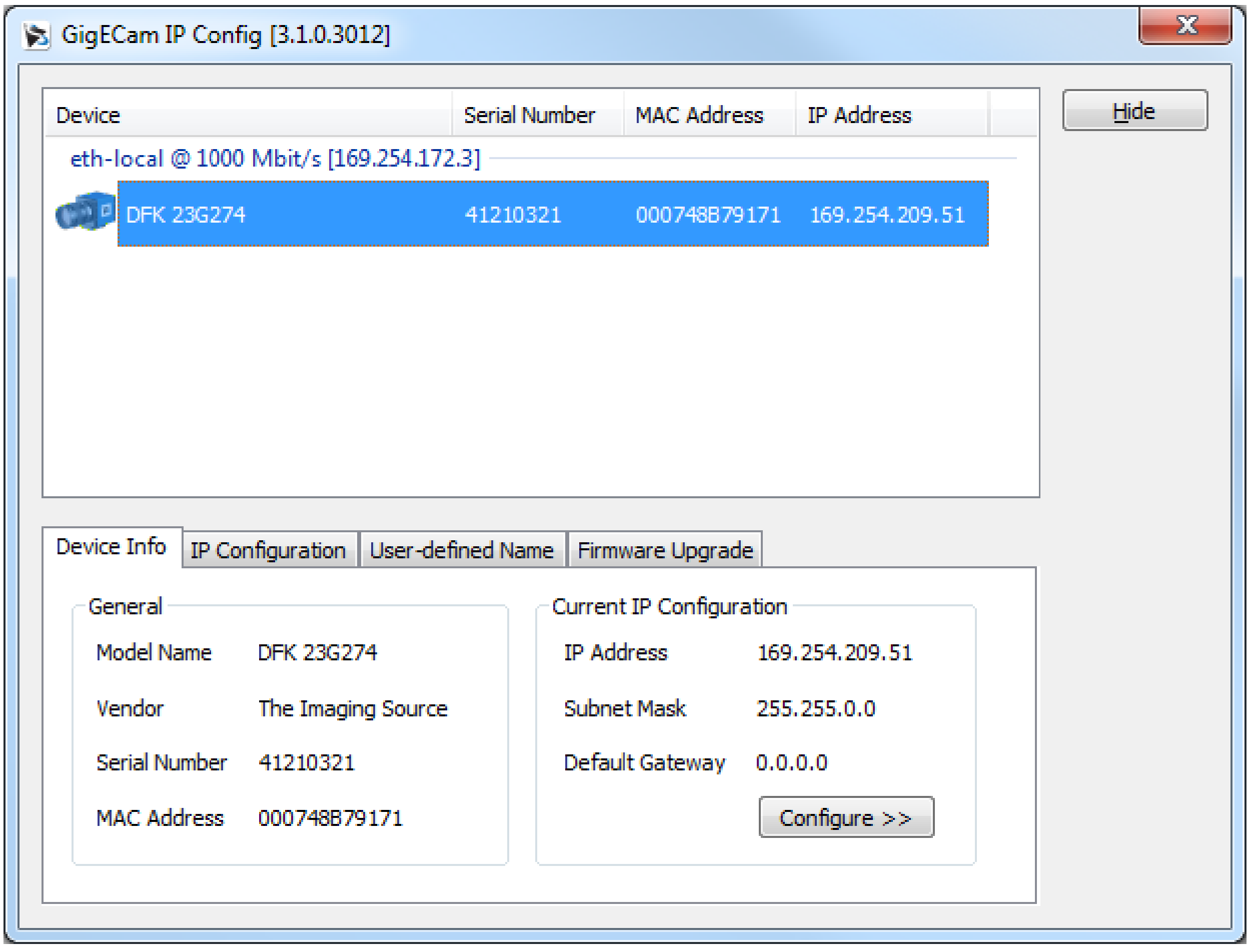
1 Description

ImagingSource cameras control plugin, based on IC Imaging Control .NET library SDK version 3.4.

https://www.theimagingsource.com/

The plug-in requires following to be installed:
- camera's drivers, can be downloaded on the Imaging Source website
- latest Microsoft Visual C++ runtime files (version 2015 can be downloaded from MS website)

Ethernet (GigE) version of camera requires to be properly configured using the "GigECam IP Config" tool The default IP configuration will be probably working, but take care of that - if there is no DHCP server, you should configure the camera manually.



Device	Serial Number	MAC Address	IP Address
eth-local @ 1000 Mbit/s [169.254.172.3]			
DFK 23G274	41210321	000748B79171	169.254.209.51

General

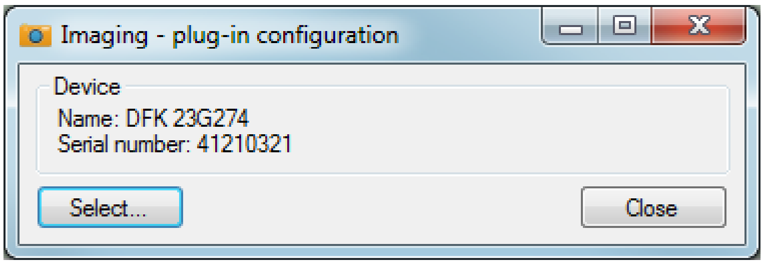
Model Name	DFK 23G274
Vendor	The Imaging Source
Serial Number	41210321
MAC Address	000748B79171

Current IP Configuration

IP Address	169.254.209.51
Subnet Mask	255.255.0.0
Default Gateway	0.0.0.0

GigeECam IP Config tool

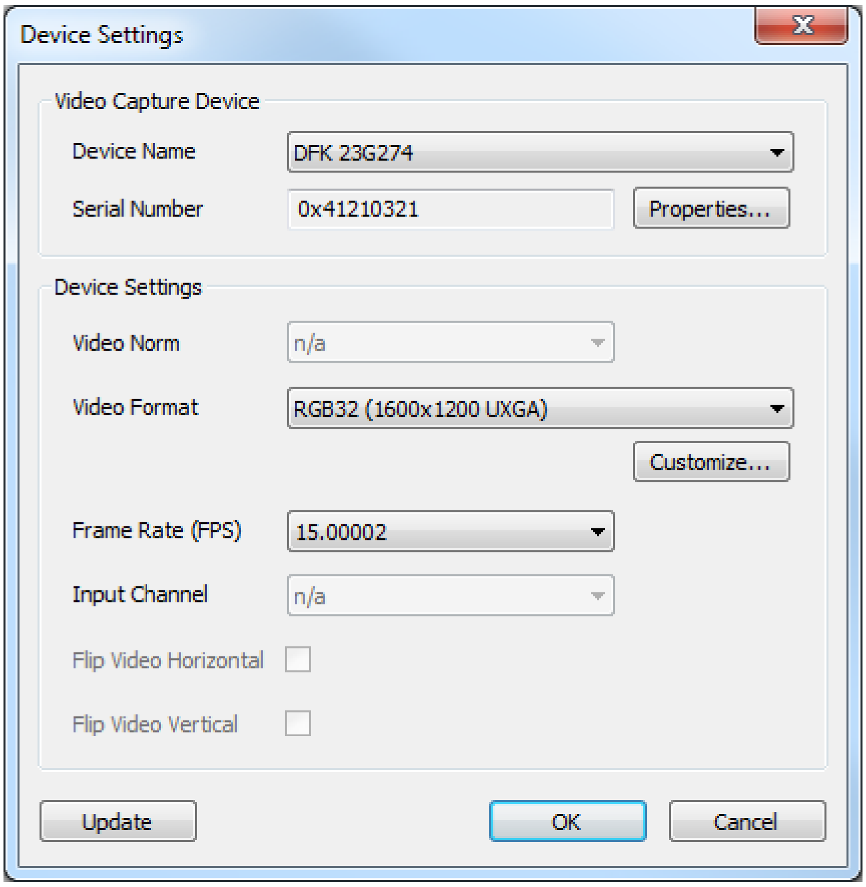
In the plugin's configuration there is no difference between USB/GigE or firewire camera, just select the desired camera.



Imaging - plug-in configuration

Device
Name: DFK 23G274
Serial number: 41210321

Plug-in configuration



Device Settings

Video Capture Device

Device Name	DFK 23G274
Serial Number	0x41210321

Device Settings

Video Norm	n/a
Video Format	RGB32 (1600x1200 UXGA)
Frame Rate (FPS)	15.00002
Input Channel	n/a
Flip Video Horizontal	☐
Flip Video Vertical	☐

Camera selection

2 Commands

2.1 *idn? (Identification)

```
*idn?
```

Gets the plug-in identification string.

Parameters

No parameters.

Return value

The identification string in standard format "`<company>, <product/name>, <serial-no>, <version>`".

2.2 load (Load definition)

```
load:<filename>
```

Runs the currently loaded definition of inspection. Blocking method, results are available through the res? command.

Parameters

`filename`	[string]	Path to the file with definition of inspection.

Return value

No return value

2.3 run (Run inspection)

run{:<section$_0$>:...:<section$_N$>}

Runs the currently loaded inspection using selected or all sections. Blocking method, results are available through the res? command.

Parameters

`section`	[string]	Section(s) of inspection to run. **Default:** if none entered, all inspection will be run

Return value

Number of launched sections.

2.4 res? (Query inspection results)

```
res?:<return-name>{(<arg-name>)}{:<return-name2>{(<arg-name2>)}}{:...}
```

After inspection was successfully ran, using this command it is possible to read results of each inspection.

Parameters

`return-name`	[string]	Inspection's result, defined by its 'return' definition key. At least one return is required to read results.
`arg-name`	[string]	An optional argument of concrete return, to get only a part of return value.

Return value

Depends on inspection type:
- **#color** - `r:g:b:h:s:i:bright:accept`

Example

Note: the 'y0' and 'y1' are a #color inspections.

`res?:y0`
Returns all values of y0 inspection in format "`r:g:b:h:s:i:bright:accept`".

`res?:y0(bright)`
Returns only "`bright`" value of specified result.

`res?:y0(bright):y1(bright):y1(r)`
Returns "bright" values from both #color inspections and "r" value from y1 inspection in format "$\texttt{bright}_{y0}$:$\texttt{bright}_{y1}$:$\texttt{r}_{y1}$".

2.5 res (Save results)

```
res:save:<filename>
res:snap:<filename>{:overlay=[bool]}
```

Save last inspection results to a text-file (`res:save` command) or save last captured and inspected image (`res:snap`) to a file with optional overlay drawing from the definition. Inspection must be ran before this command can be executed.

The text-file of inspection results has an INI format, i.e.:

```
# ImagingControlPlugin Results
# 10.12.2013 12:49:21

[y0]
r=123
g=200
b=15
bright=113
spread=81

[y1]
...
```

Parameters

`filename`	[string]	Target filename: • text file (with any extension) for `res:save` and • image file for `res:snap` (file type determined by extension, possible are .bmp, .png or .jpg/jpeg).
`overlay`	[bool]	If true, the overlay layer from definition is drawn on the image before saving. **Default:** false

Return value

No return value.

Example

`res:save:"c:\\results.txt"`
Saves result of last inspection to an INI-like text file.

`res:snap:"c:\\image.jpg"`
Saves the captured image before the inspection was ran to a JPEG-format file.

2.6 snap (Snap and save current capture)

```
snap:<filename>
```

Snap and saves the currently captured image from the camera to a file.

Parameters

`filename`	[string]	Target filename, file type determined by extension, possible are .bmp, .png or .jpg/jpeg.

Return value

No return value.

Example

```
snap: "c:\\snap.png"
```
Saves the currently captured image a PNG-format file.

3 Inspection definition

Inspection is declared by a text-representation. The representation contains **definition items**. Definition item can be of following type:
- **visual regions** (source for inspections)
- **inspections** (starting by hash-mark '#')

Format of definition item

```
item-type {
      arg_0: value_0;
      arg_1: value_1;
      ...
      arg_N: value_N;
}
```

`item-type`	Specifies the type of visual region or inspection type. Inspections starts by a hash-mark '#'.
`arg`	Argument name of concrete item. Depends on item type.
`value`	Value of specified argument.

Every definition item (region or inspection) type has its own required arguments.

Always required arguments

Visual regions
- `name [string]` - name of source region

Inspections
- `source [string]` - name of source visual region to be used by concrete inspection
- `return [string]` - name of target variable in the result set

3.1 Source regions

3.1.1 rect (Rectagular region)

Defines a rectangular region by x, y coordinates and size.

Arguments

`x, y`	[int]	Location of region in [px].
`w, h`	[int]	Size of region (width and height) in [px].

Example

```
rect {
      name: rect0;
      x: 200;
      y: 150;
      w: 100;
      h: 75;
}
```

Rectangle with size of 100x75 px at location 200, 150.

3.1.2 circle (Circular region)

Defines a circular region by x, y coordinates of center and radius.

Arguments

`x, y`	[int]	Center of circle in [px].
`r`	[int]	Radius of circle in [px].

Example

```
circle {
      name: circle0;
      x: 200;
      y: 150;
      r: 50;
}
```

Circle with center at (x,y) 200,150 and radius of 50px (diameter 100px).

3.1.3 ring (Annulus region)

Defines an annulus region by x, y coordinates of center and inner/outer radius.

Arguments

`x, y`	[int]	Center of ring in [px].
`r-inner, r-outer`	[int]	Inner and outer radius of ring in [px]. Outer circle must have a larger radius than inner circle.

Example

```
ring {
      name: ring0;
      x: 200;
      y: 150;
      r-inner: 50;
      r-outer: 75;
}
```

Ring with center at (x,y) 200,150 and inner radius of 50px and outer radius of 75px (= 25px ring size/width).

3.2 Inspections

3.2.1 #color (Color inspection)

Calculate RGB, HSI and brightness values within the specified region. The brightness is calculated simply like an average of RGB values.

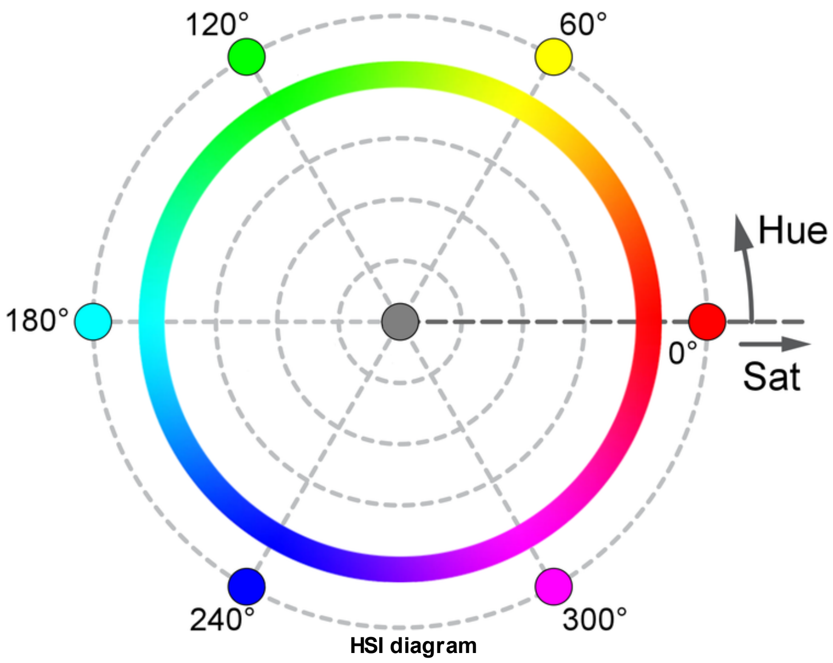
Arguments

`accept-low` (optional)	[int]	Filter pixels that have **all of** RGB values below specified. **Range:** 0 to 254 **Default:** 0
`accept-high` (optional)	[int]	Filter pixels that have **any of** RGB values above specified. **Range:** 1 to 255 **Default:** 255

Return

`r: g: b: h: s: i: bright: accept`
- **R, G, B** are floating-point numbers in range from 0 to 255 (from filtered pixels by accept-low and accept-high arguments, the same is used for HSI below)
- **H** (Hue) is floating point number representing color of degrees in range 0 to 359.999 (see the HSI diagram below)
- **S** (Saturation) and **I** (Intensity) are floating point numbers in range 0 to 1
- **bright** is floating point of brightness number from 0 (minimum brightness - black only) to 255 (maximum brightness - white only), this is similar to "I" of HSI, but accepting only accept-low argument
- **accept** is integer number from 0 to 100, reflecting percentage of pixels as result of filtering by accept-low and accept-high (i.e. 75 means that 25% pixels of region was discarded)

HSI description



HSI diagram

Source: https://en.wikipedia.org/wiki/HSL_and_HSV

Example

```
#color {
      source:  ring0;
```

```
        return: y0;
        accept-low: 5;
        accept-high: 250;
}
```

Performs a color inspection on "ring0" source region, ignoring very dark pixels (accept-low = 5) and probably over-saturated pixels (any of R, G, B over 250).