rcsCameras Documentation

Release 1.1.0

Gabriele Giovanetti

Apr 03, 2025

Contents

Contents:

CHAPTER 1

rcsCameras Package

The package rcsCameras aims at integrating in Karabo camera devices which are controlled by means of Astrotech Remote Camera Server (RCS). The RCS consists of an industrial PC (IPC) where camera vendor's API are mapped to a REST server.

1.1 Configuring the IPC

If you have a new IPC or want to relocate one, please read *How to configure a new IPC*

1.2 Power up sequence

The camera *must* be powered when the IPC is booted. In case of doubt about who came up first, just reboot the IPC.

1.3 Additional Information

- **Device classes**: *RcsGenericCamera*, *AndorZylaCamera*, *ShimadzuHpvxCamera*, *AndorIkonCamera*
- **Latest deployed version** of rcsCameras package: 1.1.0-2.6.1-zyla-test@SPB
- **Known interdependencies of this device to other services**: NTP server
- **Deployment guidelines**: Usual deployment sees a dedicated ethernet i/f (with adequate speed w.r.t expected image data throughput) patched to the RCS IPC. An NTP server must be available on the control host, and the IPC NTP client must point to it.
- **Expert contact**: Gabriele Giovanetti (Controls), Janusz Szub (ITDM) for control host setup.

CHAPTER 2

RcsGenericCamera

This Karabo Device class makes use of self description provided by RCS to generate at runtime a fully functional instance.

Please read *rcsCameras Package* documentation for general information about the hardware configuration.

2.1 Main limitations of generic devices

1. Configuration cannot be saved in Karabo project database
2. It is not possible to save images to DAQ
3. Enum parameter are not mapped to their meaning.

2.2 Known issues (may be improved in further versions)

1. Nodes parameter (currently used only by Andor Ikon camera) values are written to the HW only when the corresponding *apply* button is pressed. Therefore in the interval between their application and the write to HW their value might be reset by polling.

CHAPTER 3

AndorZylaCamera

It's the Karabo device class to control an Andor Zyla camera through an RCS IPC.

Please read *rcsCameras Package* documentation for general information about the hardware configuration.

Please refer to *Troubleshooting* section for troubleshooting.

Please read `Andor SDK 3` and `Andor Zyla HW user guide` for further reference.

CHAPTER 4

AndorIkonCamera

It's meant to control and Acquire an Andor Ikon camera. It's currently in a mocked up state. It is able to save image to DAQ. Apart from that, it is sharing limitations and issues described in *RcsGenericCamera* .

Please read *rcsCameras Package* documentation for general information about the hardware configuration.

Please read `Andor SDK documentation` and `Andor iKon-M user manual` for further reference.

CHAPTER 5

ShimadzuHpvxCamera

This is the Karabo device class to control a Shimadzu HPV-X2 camera through an RCS IPC*[0].

The main feature of this camera is the capability of acquiring a sequence of 128 (or 256) frames up to a MHz acquisition rate. The burst of images is recorded locally on the camera, then played back at a later stage.

Please read *rcsCameras Package* documentation for general information about the hardware configuration.

Please refer to the vendor's user manual for further reference.

5.1 Known Issues and Workarounds

1. The functioning of the camera implies that the multi-frame images are delivered on the karabo device output channels with a huge (> ~10s) delay. This is way beyond the usual maximum accepted delay of the DAQ Data Aggregator. For this purpose a dedicated data aggregator has been prepared and must be used. The data aggregator has been set with 15 s maximum delay. Such a delay can be easily exceeded by configuring the multi-frame acquisition.

The parameters that affect this delay are (summing up their effect): * The *recordingSpeed* parameter (which is the inverse of the frame rate whithin the

bursts).

- The number of frames in the burst (128 or 256).
- The duration of the burst recording, which equals to *recordingSpeed* x 128 (256).
- The *streamingRate* parameter (which is the rate at which the recorded burst is played back). Empirically, the value of 33 Hz for this has been proven the maximum working without losses. This means an overhead of ~ 4 s for 128 frames bursts and ~ 8 for 246 frames ones.
- There is a fixed overhead on top of that (whose major component has to do with the acquisition hardware) of the order of few seconds.

The train ID of the image burst is assigned as the one of the first frame, based on the operating system time of the Windows IPC. That's why the delay may also be affected by a poor NTP synchronization of the Windows IPC.

[0] For this camera the IPC is running a windows OS.

For reason that are not yet understood, it has been observed that the delay may increase over time with respect to the one observed after a fresh reboot of the Windows machine.

CHAPTER 6

Troubleshooting

1. If IPC is booted without camera connected (or powered), RCS software shows the parameter of a simulated camera. This is very useful for developers, but might be confusing for users. Please verify that the CameraModel parameter is present and does not read 'SIMCAM ...' (a sensible value may look like 'ZYLA5.5B-FO'). If that's not the case, first try to hit the 'Reset Camera Controller' button, which resets the IPC, then wait at least two minutes (look at 'Status' parameter for the countdown). If this still does not help, try power-cycling the camera itself (and reboot the IPC again).
2. If frame rate is lower than expected during DAQ acquisition, please verify that DAQ output channel 'hostname' is configured with the IP of the control host ethernet i/f meant to be used for DAQ connection
3. There might be some case when the camera gets to a state where a power cycling is needed anyway.
4. Invalid parameter settings: the Karabo devices do not prevent the user from entering settings that might be refused from the hardware. Such values are overwritten by polling.
5. [For controls colleagues only] Lots of useful information can be gathered by setting logger level to 'DEBUG' and looking at server output e.g. by using karabo-xterm

CHAPTER 7

How to configure a new IPC

At XFEL, the IPC is usually patched directly to a dedicated ethernet interface of a defined control host. Hence it must be configured to use a fixed IP on that subnet. To configure it for the first time (or if you want to interface it to another control host), please use the following procedure:

1. boot the IPC with a keyboard and Monitor plugged
2. login with user 'ops' and password 'Xfel12##'[1]
3. IPCs are not updated on a regular basis, so different IPCs might run different operating systems.

   A. **On older operating systems** (e.g. Ubuntu 16)

   Edit the Network Configuration file /etc/network/interfaces (e.g. with 'sudo nano /etc/network/interfaces') so that the section relative to the primary network interface reads something like (please replace `XXX.XXX.XXX.XXX` with the desired IP address for the IPC[2] and `YYY.YYY.YYY.YYY` with the appropriate subnet mask, e.g. `address 10.253.10.226` and `netmask 255.255.255.240`):

```
auto enp0s31f6

#iface enp0s31f6 inet dhcp

iface enp0s31f6 inet static

address XXX.XXX.XXX.XXX

netmask YYY.YYY.YYY.YYY
```

   B. **On newer operating systems** (e.g. Ubuntu 20 and following)

   Edit the Netplan Configuration file /etc/netplan/01-netcfg.yaml (e.g. with 'sudo nano /etc/netplan/01-netcfg.yaml') so that the section relative to the primary network interface reads something like (please replace `XXX.XXX.XXX.XXX` with the desired IP address and `YY` with the network bits defining the subnet, e.g. `10.253.10.226/28` IPC[2]):

[1] User and password may vary: if that's the case please ask vendor.
[2] The IP address and subnet should be indicated by ITDM

```
network:
 ethernets:
   enp0s31f6:
       dhcp4: false
       addresses:
           - XXX.XXX.XXX.XXX/YY
```

then reboot the machine, e.g. with *sudo reboot*.

After that is done, the NTP client on the IPC needs to be configured accordingly [See *NTP client Configuration*]. However, this can be done remotely, logging on the IPC from the control host.

NOTE: of course, this procedure applies only to Linux IPC (e.g. Andor cameras).

CHAPTER 8

NTP client Configuration

Camera images are timestamped on the RCS IPC, which need its NTP client to be configured to have access to a valid NTP server. In the typical deployment the industrial PC is patched to a specific control host. Thus an NTP server must be set up on the control host and the ntp client must be configured accordingly on the IPC:

A. **On older operating systems** (e.g. Ubuntu 16) Edit the file `/etc/ntp.conf` file on the industrial PC: it must contain an entry like

```
server xxx.xxx.xxx.xxx
```

B. **On newer operating systems** (e.g. Ubuntu 20 and following) Edit the file `/etc/systemd/timesyncd.conf`: Edit the `NTP` line in the `[Time]` section (and uncomment it if needed), so that it looks like:

```
[Time]
NTP=10.253.15.225
```

(being `xxx.xxx.xxx.xxx` the IP address of the patched ethernet i/f on the control host).

Please refer to `RCS User Manual` for details about RCS IPC administration.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search