

# ImagePro Device Server User's Guide (V3.2)

11th February 2002

*A. Götz*



## Abstract

This guide describes how to install the ImagePro device server on Windows (95/98/NT/2000), the commands implemented in the TACO ImagePro device server and how to use it from SPEC.

## 1 Introduction

ImagePro is an commercial product for doing image acquisition and analysis on Windows 98/NT. It works with a large number of CCD cameras e.g. Frelon, Princeton, Photonics, Sensicam, Photometrics and supports a rich set of image analysis routines. A useful feature of ImagePro is its macro language for automating acquisition and analysis. The TACO ImagePro device server has been written in order to allow SPEC or any other TACO client to trigger ImagePro commands remotely.

## 2 Getting started

In order to run the device server you need to do the following :

1. install Imagepro 4.0, 4.1 or 4.5 from the Imagepro CD
2. get the imagepro.zip file (from /segfs/dserver/classes/ccd/imagepro) und unzip it in e.g. c:\Taco
3. copy the oncrpc.dll to the Windows system directory (e.g. c:\WINDOWS) or add the directory C:\Taco\lib to the PATH environment variable.

4. define an imagepro device in the TACO database e.g. ID11/IMAGEPRO/1 and load the ImagePro device server commands in the database (they are in C:\Taco\res\ImagePro.res)
5. Create shortcuts to the portmapper (in C:\Taco\bin\portmapper.exe) and the imagepro device server (in C:\Taco\bin\imagepro.exe). Use Properties on the shortcuts to set the portmapper to come up as an iconized window and imagepro to start with the device server name specified in the TACO database.
6. start ImagePro (by clicking on the “ImagePro” shortcut)
7. start the portmapper (by clicking on the “portmap” shortcut)
8. start the device server with the correct personal name e.g. id11 (by clicking on the “imagepro” shortcut)
9. if you are using the Frelon with the EDT card make sure the EDT card is correctly configured for the Frelon (1024 or 2048) resolution and depth (14 or 16 bit) by executing the command :

```
“initcam -u 0 -f c:\edt\pdv\camera_config\frelon1024mux.cfg”
```

or

```
“initcam -u 0 -f c:\edt\pdv\camera_config\frelon2048mux.cfg”
```

Now you can send commands to ImagePro by calling the ImagePro device server. The device server works with version 3.x and 4.x of ImagePro 32 bit but the 16 bit ImagePro for Windows 3.1 and DOS will not work !

## 3 Database

The ImagePro device server has only one resource it reads from the TACO database :

camera\_type - 1 for Frelon or undefined for all other cameras

Define this resource only if you intend using ImagePro with the Frelon.

## 4 SPEC

The main client of the ImagePro device server at the ESRF is SPEC. There are different ways of using the ImagePro device server from SPEC.

### 4.1 SPEC CCD mode

The latest version of the TACO ImagePro device server (V3.x and later) supports the SPEC CCD device model. This means the ImagePro device server looks the same as the Frelon device server or any other device server which implements the ESRF CCD standard commands. Most SPEC ccd commands

and macros like *ccdtake*, *ccdread* etc. can be used with ImagePro. Here is a list of useful commands :

1. **config** - the first step is to use the config editor to create a CCD device with the correct device name e.g. "id11/imagepro/1". Specify as "ESRF General CCD Dev" as device type. Switch protocol to TCP to avoid network timeouts.
2. **ccdmenu** - call ccdmenu first before doing any other commands. Use the menu driven interface to select remote data saving (preferred), binning etc.
3. **ccdtake t n** - use this command to take n images of t seconds (normally the settings specified in Frelon GUI interface are respected i.e. soft/hard trigger, sequence etc.)

Refer to the SPEC CCD web page for more information on the SPEC CCD macros.<sup>1</sup>

## 4.2 ImagePro CCD mode

If you need to build your own macros to trigger and take images because the standard ccd does not satisfy your needs then use the set of ImagePros macros for SPEC. These macros have been written to exploit the additional features of the ImagePro device server which are not supported by the standard SPEC CCD's e.g. running IPBasic macros, doing online data analysis etc. This macro set is part of the imagepro zip file and can be found in macro\imagepro.mac. It can be copied to ~specadm/umacros and loaded using the SPEC command "udo imagepro". Local help can be displayed on this command set using the command "help local imagepro" :

- **NAME**

Macros for using ImagePro CCD image acquisition software

- **OVERVIEW**

This macro set can be used to trigger CCD image acquisitions by ImagePro running on a Windows PC. The macros allow a user to configure the image acquisition basic setup (device, file prefix, exposure time). Triggering is switched on/off using the ipon/ipoff macro macros. When triggering is on image acquisition is triggered at every ct command i.e. during scans. Single shot images can be taking using the ipsnap command.

- **EXAMPLE**

**ipsetup id18/imagepro/1 c:\\mypath myimg** Will setup the ImagePro macros to trigger an acquisition from the device id18/imagepro/1 with file prefix "myimg". The file names created will be of the form myimg\_n\_i where n is the scan number and i is the point number inside a

---

<sup>1</sup>[http://www.esrf.fr/computing/bliss/guides/detection/ccd/ccd\\_user.html](http://www.esrf.fr/computing/bliss/guides/detection/ccd/ccd_user.html)

scan.

**ipoff** Switch triggering off at ct command

**ipon** Switch triggering on at every ct command

**ipsnap 10** Take a single shot image exposure 10 s and store it in a new workspace (NOTE: if the file exists already ImagePro will open a dialog box which has to be acknowledged)

**ipsave c:\mypath\myimage** Save the active workspace in a file c:\mypath\myimage.tif in TIF format (this macro can be useful when used in conjunction with ipsnap)

**iptake c:\mypath\myimage 10** Take a single shot image exposure 10 s and store it in the file c:\mypath\myimage.tif (NOTE: if the file exists already ImagePro will open a dialog box which has to be acknowledged)

**ipgetimage** Get the current image and store it in the shared memory short array **ip\_image[]** (the image can be displayed using the display program called *dis* - simply start *dis* and select shared memory segment *ip\_image*)

**ipnewfile** Setup the file prefix and suffix and image number for automatic file saving.

**ipautosave** Create a file name from the file prefix, image number+1 and suffix (specified in ipnewfile) and write the current image to this file

**ipmacro mymacro** Run a macro of the given name (assumes the macro is already loaded)

**ipseqsave n** Save n images from a sequence of images

## 5 Saving images

Saving images are an essential part of any image acquisition experiment. The Imagepro Frelon driver and the device server offer a number of different possibilities. Choose the one best suited to your needs :

1. **“on-the-fly”** - this mode writes the images to harddisk as they arrive from the camera. It is activated from the window called “Storage” on the Frelon GUI. Click on the HDD mode. The images will be written as you take data. The speed with which they will be written will depend on your harddisk and PC. On a Pentium 4 writing to a EIDE disk we have found that we need a minimum of 40 ms exposure time for a 2048x2048 image in order not to lose any images. As the size of the images gets smaller (using binning or roi) the file saving decreases and therefore also

the minimum exposure time. Try this on your PC to benchmark the best figure. The images are written in ESRF data format (EDF). The file names follow the ESRF naming convention (*suffixnnnn.edf* where nnnn is an auto-incremented number). You can write to local disk (20-40 MB/s depending on the speed of your local disk) or NF) mounted NICE disks (8-10 MB/s, contact [eyer@esrf.fr](mailto:eyer@esrf.fr) to install Reflection NFS for Windows) or even to another PC via SMB mounted disk (10 MB/s).

2. **“remote”** - in this mode the device server saves the images to local or remote disk (cf. comment about disk speeds above) after the image(s) has been taken. It is activated from ccdmenu or ccdnewfile or by using the DevCcdWriteFile or DevIpWsSaveAs command in the device server. The files can be saved in ESRF or tif or any flat file format supported by ImagePro. This mode supports any exposure times because it saves the images after the acquisition. In this mode you are limited by the amount of RAM you have in your PC for taking sequences of images.
3. **“spec”** - in this mode SPEC reads the image from the device server to the local workstation where SPEC is running and then saves it locally. This is the slowest mode of working because the images are first transferred via the local network from the PC to the workstation where SPEC is running and then written to disk (normally NICE).

## 6 Commands

### 6.1 DevIpMacroRun

- *description* - run a macro on ImagePro either from a file or directly from memory
- *input* - [0] “macro name”, [1] “script file” (“” to run from memory)
- *output* - none

### 6.2 DevIpMacroStop

- *description* - stop a macro running
- *input* - [0] “macro name”, [1] “mode”
- *output* - none

### 6.3 DevIpMacroLoad

- *description* - load a script file into memory
- *input* - [0] “script file name”
- *output* - none

## 6.4 DevIpAcqControl

- *description* - modify an acquisition parameter e.g. exposure time. This is a general purpose ImagePro command to change various control parameters of a camera. The exact usage of this command is camera dependant. The best way to find out which parameters can be modified and how this is done is by setting the “Record Macro” mode and then using ImagePro to execute the command. The exact format of the IPAcqControl command will be displayed in the macro window. Refer to ImagePro macro manual as well.
- NOTE : in order to make the command flexible all parameters are transferred as shorts. This means that if a long parameter has to be transferred then it must be passed as word inverted format in two words e.g. [2]=1000 and [3]=0 to pass 1000 as a long
- *input* - [0] “command”, [1] “parameter”, [2] “imagepro parameter 1”, [3] “imagepro parameter 2”
- *output* - none

## 6.5 DevIpAcqSnap

- *description* - take a single image and store it in a new workspace on the screen. This command returns immediately and changes the state of the camera to DEVRUNNING. The state changes back to DEVON when the acquisition is finished. During this time no acquisitions can be started.
- *input* - “exposure time (in milliseconds)”
- *output* - none

## 6.6 DevIpWsSaveAs

- *description* - save the current workspace in the specified file as TIF format.
- *input* -[0]=file name e.g. c:\mypath\myimage, [1]=format (e.g. “edf”, “tiff”, “flf”, ...)
- *output* - none

## 6.7 DevIpAcqTimed

- *description* - start a single or a series of timed acquisitions, this command return immediately and changes the state of the camera to DEVRUNNING. The state changes back to DEVON when the acquisition is finished. During this time no acquisitions can be started.
- *input* - [0] “directory to save images in” (“” to save in memory), [1] “file name prefix” (“” to save in memory, [2] “start number”, [3] “no. of frames”, [4] “time between frames”
- *output* - none

## 6.8 DevIpSetVariable

- *description* - pass a value to an imagepro internal variable (defined in the .INI file)
- *input* - [0] “variable name”, [1] “value”
- *output* - none

## 6.9 DevIpGetVariable

- *description* - get a value from an imagepro internal variable (defined in the .INI file)
- *input* - “variable name”
- *output* - value

## 6.10 DevIpGetImageSize

- *description* - get the current image size (rows and columns)
- *input* - nothing
- *output* - [0] “rows”, [1] “columns”

## 6.11 DevIpGetImage

- *description* - returns current image as opaque 16 bit array (note: if reading the image from a Unix box then you have to swap the array before using it using *array\_op()*)
- *input* - nothing
- *output* - image as opaque array

## 6.12 DevIpGetArea

- *description* - returns a part (or whole) of the current image as opaque 16 bit array (note: if reading the image from a Unix box then you have to swap the array before using it using *array\_op()*)
- *input* - coordinates of area, [0] “left”, [1] “right”, [2] “top”, [3] “bottom”
- *output* - area as opaque array

## 6.13 DevIpWsChangeDescription

- *description* - change the description of any one of the image description tags for the current workspace. Call this command before you save workspace with DevIpWsSaveAs.
- *input* - [0] = tag (“title”, “artist”, “date”, “description”, “name”, or “range”), [1] = description (e.g. motor settings)
- *output* - none

#### 6.14 DevCcdStart

- *description* - start CCD acquisition
- *input* - none
- *output* - none

#### 6.15 DevCcdStop

- *description* - stop CCD acquisition (does nothing for the moment)
- *input* - none
- *output* - none

#### 6.16 DevCcdRead

- *description* - read CCD image in current workspace
- *input* - unknown, ask David Fernandez (D\_LONG\_TYPE)
- *output* - image (D\_OPAQUE\_TYPE)

#### 6.17 DevCcdSetExposure

- *description* - set image exposure time
- *input* - exposure time in seconds (D\_FLOAT\_TYPE)
- *output* - none

#### 6.18 DevCcdGetExposure

- *description* - get image exposure time
- *input* - none
- *output* - exposure time in seconds (D\_FLOAT\_TYPE)

#### 6.19 DevCcdSetBin

- *description* - set horizontal and vertical binning
- *input* - [0]=horizontal binning, [1]=vertical binning (D\_VAR\_LONGARR)
- *output* - none

#### 6.20 DevCcdGetBin

- *description* - get horizontal and vertical binning
- *input* - none
- *output* - [0]=horizontal binning, [1]=vertical binning (D\_VAR\_LONGARR)



### 6.21 DevCcdSetTrigger

- *description* - set trigger
- *input* - unknown, ask David Fernandez (D\_LONG\_TYPE)
- *output* - none

### 6.22 DevCcdGetTrigger

- *description* - get trigger
- *input* - none
- *output* - unknown, ask David Fernandez (D\_LONG\_TYPE)

### 6.23 DevCcdSetFilePar

- *description* - set file parameters
- *input* - [0]=absolute root directory, [1]=prefix, [2]=suffix, [3]=image no., [4]=printf() format for image no., [5]=overwrite flag (y=true) (D\_VAR\_STRINGARR)
- *output* - none

### 6.24 DevCcdGetFilePar

- *description* - get file parameters
- *input* - none
- *output* - [0]=absolute root directory, [1]=prefix, [2]=suffix, [3]=image no., [4]=printf() format for image no., [5]=overwrite flag (y=true) (D\_VAR\_STRINGARR)

### 6.25 DevCcdWriteFile

- *description* - write a file
- *input* - the frame number (if multiple frames) (D\_LONG\_TYPE)
- *output* - none

## 7 Supported cameras

The following cameras at the ESRF are supported in ImagePro :

1. Frelon (ID11, ID2, Frelon laboratory)
2. Quantix (ID11)
3. Photonics Science XIOS II (ID22 )
4. Photonics Science FDI (ID18)
5. Photonics Science (ID13)

## 6. Medoptics (ID11)

The following cameras have ImagePro drivers but are presently not being used with ImagePro at the ESRF (it would be easy to move them to ImagePro if a beamline so desires) :

1. Sensicam (ID10, ID2, ...)
2. Matrox (ID13, BM5, ID1, ...)

For a full list of cameras for which Imagepro drivers exist refer to the ImagePro website (<http://www.mediacy.com>). If needed a driver can be written locally. Drivers are quite easy to write and can be done within a day for simple cameras.

## 8 Troubleshooting

Here are some problems you could encounter :

1. *cannot unzip imagepro.zip because WinZip is not installed on on my PC* - you can pick up a copy of WinZip at the ESRF from **ftp://ftp.esrf.fr/pub/cs/micro/winzip/win9xnt**
2. *cannot start device server, complains about TACO initialisation* - try starting the portmap first
3. *device of this name not found* - make sure you have positioned NETHOST correctly and that there is a device defined in the static database
4. *writing files to NICE is too slow* - ask CS to help you mount NICE via NFS on your PC. This way you will get performances at least as good as Unix to NICE if not better (8-10 MB/s with peaks of 16 MB/s).
5. *ImagePro creates a empty window instead of taking a real image* - this is known to happen with IPP 4.0 and 4.1 if there are too many windows open. Make sure you close your windows before snapping. A good trick is to write a macro "closeall" which you call before every snap. Apparently this bug is fixed in IPP 4.5.
6. *ImagePro does not display the image with the range* - this happens if you have a 16bit camera and a very lowlevel image. The best fit algorithm of IPP forces a minimum of 32 levels with a minimum of 16 between each level. The solution is to write your own best fit macro which will scale the image as you think best e.g. min and max, after each snap. Examples of such macro are available from the author.

## 9 Changes

The following changes have been made in this version :

### **9.1 V3.2**

1. improved support for Frelon camera and SPEC e.g. it is now possible to save sequences of images as EDF format
2. improved some of the error handling

### **9.2 V3.0**

1. added support for ESRF CCD commands so that ImagePro can be integrated seamlessly into SPEC (ImagePro is now compatible with the Unix version of the Frelon device server)

### **9.3 V2.0**

1. added support for Frelon 2000 SDK interface
2. added support for ESRF edf file format

### **9.4 V1.4**

1. added commands DevIpGetVariable and DevIpWsChangeDescription

### **9.5 V1.3**

1. added commands DevIpGetImageSize, DevIpGetImage and DevIpGetArea

### **9.6 V1.2**

1. added commands DevIpAqcSnap and DevIpWsSaveAs
2. image is displayed after acquisition before being saved (command IpAcq-Timed)
3. improved SPEC macros so they don't block, added command ipsave, changed all time arguments from milliseconds to seconds

## **10 Conclusion**

The above set of commands allow a user to start any macro or acquisition in ImagePro remotely. It is now possible to use the ImagePro device server like other ESRF CCD's in SPEC. Reading images over the network via the device server into SPEC will however be slower than displaying directly on the PC screen. Therefore users are encouraged to use the ImagePro display possibilities directly. The next step is to add the ESRF CCD compatible commands for taking sequences of images.