EYER-OP SDK
DEVELOPER MANUAL

Version 0.3

Opgal Optronic Industries Ltd.
www.opgal.com
Tel: +972 4 995-3903
Fax: +972 4 995-3900
PO Box 462, Industrial Area 5, Karmiel 2161401, Israel

EYER-OP SDK DEVELOPER MANUAL

Version	Author	Summary of changes
0.1	Sebastian Cabot	Initial Release
0.2	Sebastian Cabot	Fix duplicate command reference at the wrong document section
0.3	Sebastian Cabot	Update to include Hawkeye support Update requirement sections for latest version Improve the Configuration & Runtime sections Add documentation for the ApplyActions command Add documentation for the frame header

Opgal Optronic Industries Ltd.
www.opgal.com
Tel: +972 4 995-3903
Fax: +972 4 995-3900
PO Box 462, Industrial Area 5, Karmiel 2161401, Israel

Contents

Opgal Optronic Industries Ltd.
www.opgal.com
Tel: +972 4 995-3903
Fax: +972 4 995-3900
PO Box 462, Industrial Area 5, Karmiel 2161401, Israel

Opgal Optronic Industries Ltd.
www.opgal.com
Tel: +972 4 995-3903
Fax: +972 4 995-3900
PO Box 462, Industrial Area 5, Karmiel 2161401, Israel

Overview

About This Manual

Audience

This manual is intended for programmers wishing to interface with EyeR-OP, Hawkeye, Therm-App® or Therm-App® PRO from Linux systems.

What's in the Manual

This manual has four sections:

- Overview – This section with general information about this manual and the EyeR-OP SDK's architecture.
- Setup – Covers all the steps necessary to install the SDK on the target system.
- Configuration – Covers the different configuration options for capturing thermal images from the attached sensor.
- Runtime SDK Operation – Covers the different runtime API and utilities available for capturing data and controlling the attached sensor.
- Reference Manual – Has a reference of commands and Video Pipeline plugins as well as of the image header passed with every frame

EyeR-OP SDK Overview

Purpose

The SDK can be used to connect Opgal USB based sensors. These include the EyeR-OP embedded engine, the Therm-App® family of sensors and the Hawkeye family of sensors. These sensors connect to the host using a standard USB 2.0 port.

Architecture

The SDK uses a daemon process to connect to the sensor. The captured data is transferred to the target application using shared memory.

Controlling the capture daemon is done using a utility which allows sending messages to the running capture daemon.

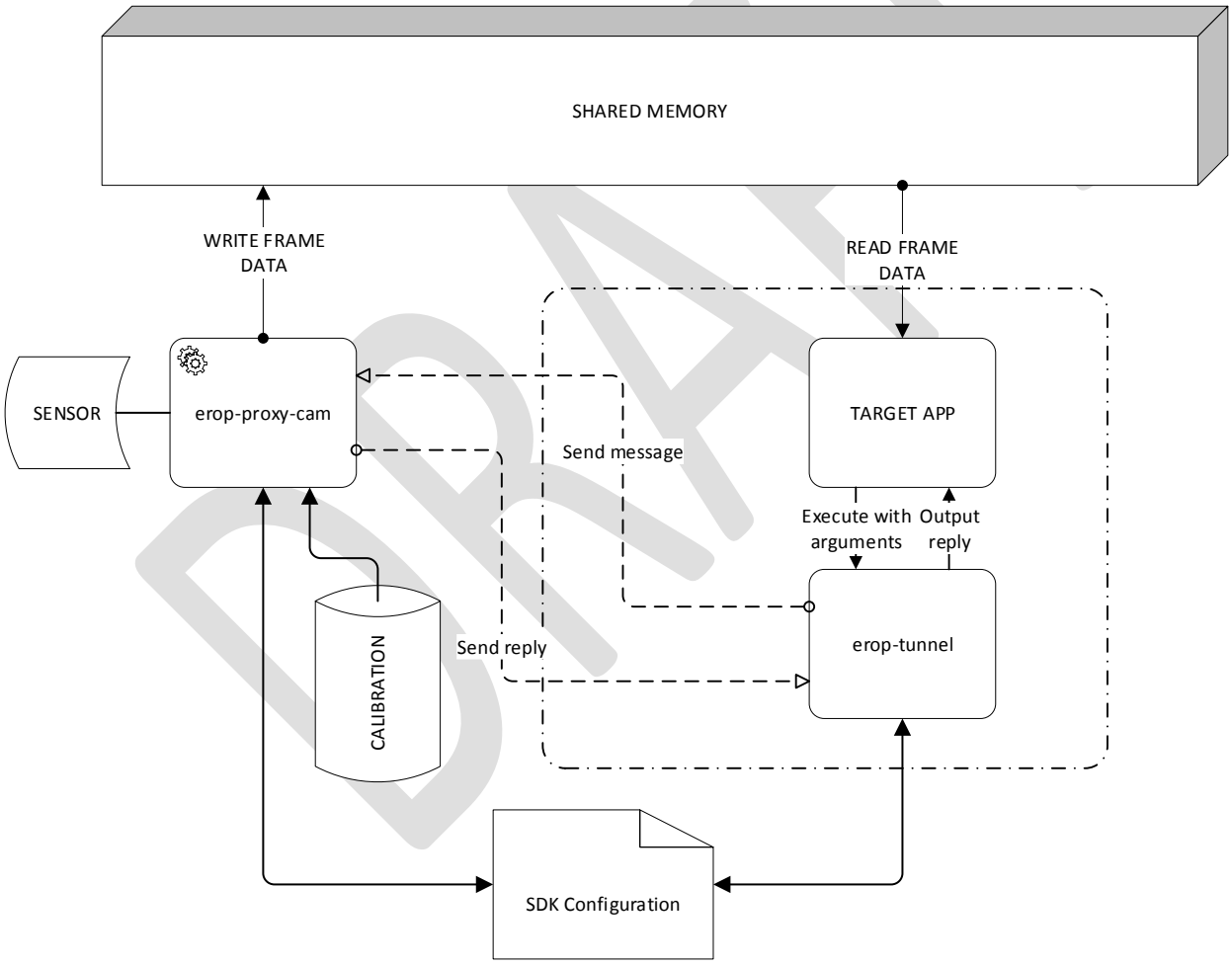

FIGURE 1 SDK ARCHEITECTURE OVERVIEW

Opgal Optronic Industries Ltd.
www.opgal.com
Tel: +972 4 995-3903
Fax: +972 4 995-3900
PO Box 462, Industrial Area 5, Karmiel 2161401, Israel

Setup

Requirements

Supported Processor Architectures

The SDK supports both 32-bit ARM and 64-bit ARM of the ARMv7-A and ARMv8-A processor families.

Please Note: ARMv7-A processors must have hardware NEON and floating-point support

Please Note: The 32-bit ARM version is optimized for Cortex-A9. The 64-bit ARM version is optimized for Cortex-A57.

Software Dependencies

The SDK can only be installed on a Linux operating system.

The following runtime components must be properly installed on the target system:

- libc6 = 2.28
- libstdc++6 = 6.0.25
- libpthread
- libusb-1.0 = 1.0.19
- libpng16 = 1.6.36

Please Note: Our SDK is compiled using GCC/G++ 8.x. It may be possible for ABI compatible versions of the GCC compiler suite and corresponding libraries to work as well. Please refer to the ABI Policy and Guidelines in the GCC documentation (*https://gcc.gnu.org/onlinedocs/libstdc++/manual/abi.html*).

Please Note: Linux NETLINK sockets must be enabled in the kernel (This is generally enabled by default)

Please Note: Make sure to obtain the correct SDK installation for your target Linux architecture.

Opgal Optronic Industries Ltd.
www.opgal.com
Tel: +972 4 995-3903
Fax: +972 4 995-3900
PO Box 462, Industrial Area 5, Karmiel 2161401, Israel

Installing the SDK

Understanding the Installation File Name

The SDK's installation archive is a gzipped tar archive. You should have received a file named *eyerop-<arch>-<version>-SDK.tgz*

- <arch> - designates the architecture of the installer in the format of <machine>-poky-linux. For instance, arm, i686 or aarch64
- <version> - The SDK version
- Hawkeye specific versions will have the HAWKEYE string replace the SDK string in the file name

Installation Procedure

Installing the SDK is really simple.

- Copy the tgz file to the target machine
- Create a directory for the installation files and extract the installation archive into it

```
root@dev-lx:/tmp$ mkdir eyeropsdk
root@dev-lx:/tmp$ cd eyeropsdk/
root@dev-lx:/tmp/eyeropsdk$ tar xzf ../eyerop- arm-1.9.0.5466-SDK.tgz
```

- Run the installation script (Please note that you need to be root to install the SDK. So, if you are not on an embedded system you will probably want to use *sudo*)

```
root@dev-lx:/tmp/eyeropsdk$ ./install.sh
```

- Once the script returns with no errors you are done. After the installation is complete you can delete the installation file and directory.

Installing Calibration Files

By this point the SDK is properly installed. However, even if the sensor is properly connected to the system no data will be acquired. To be able to start capturing images we first need to install the correct calibration files.

Although calibration files can be installed anywhere on the filesystem, unless we want to edit the configuration files we should install the files to: /opt/eyerop/calibration/ir0. The folder is not created by default so let's go ahead and create it:

```
root@dev-lx:~$ mkdir -p /opt/eyerop/calibration/ir0
```

Once the folder is created copy the calibration files into it. The calibration files have ".bin" extension.

Acquiring the calibration files depends on the product to be used with the SDK.

Calibration files or for Therm-App® should be downloaded using a mobile phone and the Therm-App® application.

If you require assistance obtaining the calibration files, please contact customer support.

Please Note: It is not possible to capture any data before installing the calibration files.

Verifying that the Installation Works

Once the installation is complete and the calibration files are in place we can proceed to check if everything works as it should.

- Open two terminal windows connected to the system (Depending on the target platform different methods exist for doing so. SSH comes to mind).
- Connect the sensor to the target machine
- On one terminal issue the following command:

```
root@dev-lx:~$ /opt/eyerop/bin/erop-proxy-cam -nproxycam-ir --perror --logmask=130
```

  The program will show informational output followed by a line for each captured frame with a frame counter and the frame id from the sensor
- On the second terminal issue the following command:

```
root@dev-lx:~$ /opt/eyerop/bin/ipc-cap /ipc0
```

  A line will be printed for each frame received over shared memory from erop-proxy-cam
- Both programs can be closed using CTRL+C or by issuing SIGTERM

Please Note: These programs must be executed as root

The SDK programs log their output using syslog. The --logmask and the --perror arguments determine the logging scheme that will be used.

If the --perror flag is present, then the output of the application will write its output to the console as well as to the system's log.

The --logmask argument follows syslog's setlogmask() implementation. The number supplied to the --logmask argument is used to override the default log mask for the application. The mask is specified in octal. The default value is 030 which will print errors and warnings. For printing informational messages add octal 100 to the mask. For printing debug messages add 200 to the mask. For printing all types of messages, the mask value is 330. If you wish to suppress warnings, then the mask should be 010. A mask of 000 will suppress all output.

Please Note: Although different masks in the range of 001 to 777 are valid there is really no point in specifying other mask values then the ones listed above since the application will not log to other log levels.

Troubleshooting the Installation

If you are unable to see the desired output specified in the steps above, please verify the following:

- Verify that the SDK is installed – you should have the following folders and files:
  - /opt/eyerop/bin
    - erop-proxy-cam
    - ipc-cap
    - erop-tunnel
  - /opt/eyerop/etc
    - appcommands.conf
    - eyerop.conf
    - proxycam-ir.conf → link to proxycam-irSDK.conf or to proxycam-irHK.conf
    - proxycam-irSDK.conf
    - routing.conf
    - sources.conf
  - /opt/eyerop/examples
    - ipccap

        - ipccap.cpp
        - op.cpp
        - op.h
    - python
        - querysource.py
        - switchpipeline.py
        - toggleblackhot.py
  - /opt/eyerop/scripts (For Hawkeye distributions)
    - doibit.json
    - querycaminfo.json
    - resetBit.json
    - setExternalSync.json
    - setInternalSync.json
    - setpattern.json
    - updateNuc.json
  - /etc
    - eyerop → link to /opt/eyerop/etc
- Verify that the calibration files are installed – you should have the following files and folders:
  - /opt/eyerop/calibration/ir0
    - A list of .bin files.
- Verify that the sensor is connected by issuing the *lsusb* command. In the output you should see a list of all connected USB devices. One of the lines should be similar to the highlighted line below (Please note that Bus number and Device number may be different on your system):

```
root@dev-lx:~$ lsusb
Bus …
Bus 002 Device 003: ID 1772:0002 System Level Solutions, Inc.
Bus …
```

- Verify that you execute the commands with the correct parameters
- Verify that you execute the commands as root

Configuration

Understanding the SDK's Configuration

Configuration Files Syntax

All the SDK configuration files are JSON files. This means they are easy to read and edit. Care should be taken to conform to the JSON syntax since errors in a configuration file will prevent the SDK from working. More information about the JSON syntax can be found at *http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf.*
There are various online JSON validators that can be used to verify the configuration file syntax.

We added a private extension to enable internal document links. This extension doesn't violate the JSON syntax specification.

We use links to eliminate verbose copying of values from object to object. Using links, we can define a property once and use it many times within the document. Here is an example:

```
"nuc": {
        "Type": "filter",
        "LinkMethod": "Direct",
        "ClassName": "TA_nv_nuc_bpr_16",
        "Lib": "",
        "Configuration": {
           "CalibType": "@:/Global/CalibType"
        }
}
```

In this example we can see that the property "CalibType" under nuc/Configuration links to a property defined under the root of the document within the object "Global" and has the name "CalibType".

Links must always be specified using the full path. The link designation is the "@:/" prefix. All three characters must be present in the link definition. The '/' is taken to refer to the root of the document.

Understanding the Configuration Loading Process

After first installing the SDK all the configuration files are stored under /opt/eyerop/etc. The installation process also creates a link to this folder under the system's /etc folder with the path /etc/eyerop.

The files under /opt/eyerop/etc contain the default configuration for the SDK.

The SDK is shipped with the following configuration files:

- appcommands.conf – This file lists all the command known to the processes of the SDK, their type and the permission mask needed to execute the command
- eyerop.conf – This file list all known configuration namespaces. Each configuration namespace may have an accompanying configuration file. If a configuration namespace has an accompanying file, then this file is loaded under that namespace
- proxycam-ir.conf – This is a link to the actual configuration that will be used for the erop-proxy-cam capture daemon
- proxycam-irSDK.conf – This is the regular SDK configuration file for the erop-proxy-cam daemon

- proxycam-irHK.conf – This is the Hawkeye SDK configuration file for the erop-proxy-cam daemon
- routing.conf – This file contains the necessary information for erop-tunnel to be able to forward commands to erop-proxy-cam
- sources.conf – This file contains the list of imaging source names. The default file only contains proxycam-ir.

Although the default configuration is stored under /opt/eyerop/etc if you use erop-tunnel to change the configuration of a running erop-proxy-cam instance the changes will be stored in a file under $HOME/.eyerop/. A shadow file with the same name as the original will be created under this folder. It is important to understand that erop-proxy-cam will never change the default configuration so reverting to the defaults is as simple as deleting $HOME/.eyerop/.

Please Note: The default installation installs everything with under the name eyerop. It however is possible to change this: Both erop-proxy-cam and erop-tunnel can receive an argument --sysname=<sysname>.

If you decide to use a different name than please follow the guidelines below:

- You can move (or copy) the files under /opt/eyerop to a location of your choosing, for the purpose of these instructions I will assume the new location is /opt/myapp and that the desired sysname is myapp
- Change the name of /opt/myapp/etc/eyerop to /opt/myapp/etc/myapp
- Create under /etc a link: ln -s /opt/myapp/etc myapp
- Run erop-proxy-cam and erop-tunnel from the new location and add the argument: --sysname=myapp:

```
root@dev-lx:~$ /opt/myapp/bin/erop-proxy-cam -nproxycam-ir –logmask=130 –syaname=myapp &
```

  The above command will run erop-proxy-cam in the background from the new location with the new sysname.
- The user configuration will be stored under $HOME/.myapp
- It is possible to run several "sysnames" under the same system.

Understanding erop-proxy-cam and how to configure it

The capture daemon erop-proxy-cam is a generic capture/control process for different sensors compatible with the SDK. The specific way that the daemon will work is determined by the specific Video Processing Pipeline the erop-proxy-cam daemon is configured to load. The supplied proxycam-ir.conf defines several pipelines that can be used by the erop-proxy-cam daemon. These pipelines can be edited, and other pipelines can be added.

When erop-proxy-cam is executed it will load the configuration file that matches the argument passed with the -n/--name option. You may create as many configuration files as you want and call as many instances of erop-proxy-cam as you like as long as each refer to its own configuration. However, please note that there might be various hardware and performance issues preventing you from doing so. For more information please refer to the *Using erop-proxy-cam to Connect to Multiple Sensors* section below.

A configuration file for erop-proxy-cam contains two types of configurations:

- General daemon configuration like which Video Pipeline to load

- Specific Video Pipelines distinct configurations according to the pipelines the daemon is required to support.

Although erop-proxy-cam will load even if no "Video Pipeline" is specified it will obviously not be able to do anything useful.

The anatomy of a Video Processing Pipeline

The SDK implements a generic Video Processing Pipeline framework similar in concept to GStreamer and DirectShow albeit extremely more lightweight and fast. Under this framework several plugin filters were developed to support various image processing models.

Please note: At this time Opgal doesn't expose the Video Pipeline generic framework and all supported plugins were statically linked into the daemon.

Each Video Processing Pipeline must have at least two elements – a source and a sink. A SOURCE element is responsible for acquiring the data and a SINK element may be used to transfer the data out of the pipeline once the processing is done. The video flows from SOURCE to SINK.

Between a SOURCE and a SINK zero or more FILTER elements may be added to do the actual video processing.

The Video Processing Pipeline framework supports four other kinds of pipeline elements:

- CAPTURE FILTER – Designed to do line processing for low latency in capture processing. There can be just one CAPTURE FILTER in the pipeline. If a CAPTURE FILTER exists in the pipeline, then it must be placed second right after the SOURCE element.
- TAP – A TAP is a special element which can be used to probe the data going through the pipeline at a certain point. There can be any number of TAP elements in the pipeline.
- CONTROLLER – A CONTROLLER is a special element designed to change the state of the pipeline or the system in response to an event or a request. There can be any number of CONTROLLERS in the pipeline. Please note that the SDK doesn't expose any CONTROLLER elements at this time.
- ROI FILTER – A ROI FILTER is used to do some processing on the incoming image

The Video Pipeline is executed in its own thread. With each iteration the pipeline thread requests an input from the SOURCE and propagates in turn to each FILTER until the SINK is reached.

The SOURCE element generally has its own capture thread. It is important to note the CAPTURE FILTER will run within the context of the SOURCE's capture thread if one exists. It may also be possible to have a CAPTURE FILTER present in the pipeline that will not execute because the SOURCE doesn't support it.

Each ROI FILTER and the SINK element can be connected to the previous element in the pipeline with one of three methods which are:

- Direct – zero copy of output buffers from the previous element, executing within the same thread as the previous element.
- Threaded – zero copy of output buffers from the previous element executing within a new thread. Downstream elements will also execute within this thread (unless connected as Threaded or ThreadedCopy).

- ThreadedCopy – copy memory of output buffers from the previous element executing within a new thread. Downstream elements will also execute within this thread (unless connected as Threaded or ThreadedCopy).

Please Note: The CAPTURE FILTER always executes within the context of the capture thread. TAP elements are always executed within the context of the previous element.

Please Note: The defined Video Pipeline configurations within proxycam-ir.conf are already tuned for best performance.

Understanding a Video Pipeline configuration

Below we can see the configuration of the "NucBprOnly" pipeline taken from proxycam-ir.conf. This is the default pipeline the SDK will load after installation.

```
"NucBprOnly": {
    "PipelineStructure": [
      "source",
      "adr",
      "atdf",
      "nuc",
      "ipc0",
      "sink"
    ],
    "source": "@:/Filters/source",
    "adr": "@:/Filters/adr",
    "atdf": "@:/Filters/atdf",
    "nuc": "@:/Filters/nuc",
    "ipc0": "@:/Filters/ipc0",
    "sink": {
      "Type": "sink",
      "LinkMethod": "Direct",
      "ClassName": "Null_sink",
      "Lib": "",
      "Configuration": {
        "LogIncomingFrames": true
      }
    }
  }
```

Each pipeline definition must contain a property named "PipelineStructure". This property is an array with the names of the elements in the pipeline. The order of the elements in the array is important since this will be the order in which the elements will be added into the pipeline.

Please Note: The only mandatory elements in any given pipeline are the SOURCE and SINK elements. In the example above the SOURCE element is trivially named "source" and the SINK element is trivially named "sink" but any names can be chosen as long as they are unique within the pipeline definition.

For each element listed in the array we should have a corresponding element configuration. In the example above, we can see that all elements except the sink link to a configuration object defined elsewhere in the document using our "link" notation – See the **Configuration Files Syntax** section for more details.

The "sink" element however has a full in place specification.

Opgal Optronic Industries Ltd.
www.opgal.com
Tel: +972 4 995-3903
Fax: +972 4 995-3900
PO Box 462, Industrial Area 5, Karmiel 2161401, Israel

Going over the definition of the "sink" element we can see that it is of "Type": "sink" and that it is linked to the previous element "Direct"-ly which means that buffers will be passed to it without copying and that it will be executed within the thread the previous element is executed within.

We can also see that the "Class" that implements the sink is of type "Null_sink" which is designed to discard the incoming data. And we can see that the class will be loaded from a library specified by the "Lib" property which happens to be empty and that means that "Null_sink" is statically linked to the daemon.

"Null_sink" is passed a "Configuration" object which has one property named "LogIncomingFrames" that is set to true. This property instructs the element to output a line for each valid frame that it receives.

Please Note: All pipeline elements delivered with the SDK are statically linked to the daemon so "Lib" will always be an empty string.

Please Note: All SOURCE, FILTER and SINK pipeline elements supplied with the SDK support the "LogIncomingFrame" property and it can be used to debug problems with the pipeline. For instance, you can set this property on the SOURCE element to see if any data is captured. You can set it for both SOURCE and SINK to see that no data is dropped. It is recommended that this property be set to false when not debugging.

Selecting the Video Pipeline that will be loaded by erop-proxy-cam

When erop-proxy-cam is executed it can be instructed to load a Video Pipeline and start executing it. Although it is possible to use erop-tunnel to select a pipeline to execute while erop-proxy-cam is already running it is usually convenient to have the daemon stat processing automatically when it is loaded.

Two properties in the configuration file control the behavior of erop-proxy-cam at startup: "AutoStartVideoPipeline" and "ActivePipeline". To start a pipeline at startup the value of "AutoStartVideoPipeline" should be set to "Always". The "ActivePipeline" property should be set to the name of a valid pipeline defined within the file. To prevent the daemon from loading a pipeline at startup please set the "AutoStartVideoPipeline" property to "Never".

Telling the Video Pipeline where to Search for Calibration Files

The erop-proxy-cam daemon instructs pipelines to look for calibration files under the path specified by the "CalibrationPath" property. The default specified in proxycam-ir.conf is "/opt/eyerop/calibration/ir0".

It is possible to change the "CalibrationPath" in runtime when switching the running pipeline using erop-tunnel. See the **Using erop-tunnel to Control erop-proxy-cam and the Video Pipeline** section below.

Please Note: Do not assume that the different elements will be able to locate the calibration files without explicitly setting the "CalibrationPath". Although it may be true for some elements this behavior is not consistent across elements and may change without notice.

Opgal Optronic Industries Ltd.
www.opgal.com
Tel: +972 4 995-3903
Fax: +972 4 995-3900
PO Box 462, Industrial Area 5, Karmiel 2161401, Israel

Configuring a Video Pipeline to Output Images to an Application

To pass captured images to an outside application the SDK provides a TAP element of "Type": "IPC_tap". This TAP element is designed to copy the captured image into shared memory where the target application can read from.

Please Note: The IPC_tap never changes the input data so that the data passed to the application is in the same format as the data output from the upstream filter.

The excerpt below taken from proxycam-ir.conf defines an element named "ipc0" of "Type": "tap" with class "IPC_tap". "IPC_tap" is configured to use a 4 image buffers. By setting the "IPC_Log2_Buffers" to 2. This property instructs the class to allocate 2^2 image buffers.

```
    "ipc0": {
      "Type": "tap",
      "LinkMethod": "Direct",
      "ClassName": "IPC_tap",
      "Lib": "",
      "Configuration": {
        "IPC_Log2_Buffers" : 2
      }
    }
```

This element can be placed at any place between the SOURCE and the SINK or between the CAPTURE and the SINK to tap into the buffers generated by the previous element in the pipeline.

Below you can see a pipeline definition that will output the unprocessed raw data after time domain filter:

```
  "RawCapture": {
    "PipelineStructure": [
      "source", "adr", "atdf", "ipc0",  "sink"
    ],
    "source": "@:/Filters/source",
    "adr": "@:/Filters/adr",
    "atdf": "@:/Filters/atdf",
    "ipc0": "@:/Filters/ipc0",
    "sink": {
      "Type": "sink",
      "LinkMethod": "Direct",
      "ClassName": "Null_sink",
      "Lib": "",
      "Configuration": {
        "LogIncomingFrames": false
      }
    }
  }
```

And in the next sample you can see how we use links to reuse the "ipc0" definition to define a pipeline with two taps that can get frame after NUC+BPR and after dynamic range compression.

```
"NucBprDrcYUV420": {
  "PipelineStructure": [
    "source",
    "adr",
    "atdf",
    "nuc",
    "ipc0",
    "normalizer",
    "drc",
    "ipc1",
    "sink"
  ],
  "source": "@:/Filters/source",
  "adr": "@:/Filters/adr",
  "atdf": "@:/Filters/atdf",
  "nuc": "@:/Filters/nuc",
  "normalizer": "@:/Filters/normalizer",
  "drc": "@:/Filters/drc",
  "ipc0": "@:/Filters/ipc0",
  "ipc1": "@:/Filters/ipc0",
  "sink": {
    "Type": "sink",
    "LinkMethod": "Direct",
    "ClassName": "Null_sink",
    "Lib": "",
    "Configuration": {
      "LogIncomingFrames": false,
      "ForceImageFormat": "YUV420"
    }
  }
}
```

Note that the "Null_sink" is configured to request "YUV420" images from the preceding processing element which is "drc" (taps are not considered processing elements since they never change the incoming buffer). This means that an application listening to the output of ipc1 will get YUV420 frames (without this setting the output would have been GRAY16 with only the least significant byte set with a valid value). See the

Video Pipeline Elements Reference section below for more information supported elements configuration options.

Runtime SDK Operation

Using erop-proxy-cam to Connect to Multiple Sensors

It is possible running multiple instances of erop-proxy-cam each against a separate sensor connected through USB. To do so please adhere to the following guidelines:

- Make sure each sensor is connected to a different USB bus on the target machine
- Create a new configuration file with a distinct name for each instance you wish to run
- Make sure each IPC tap you use has a distinct name across all proxy configurations
- Set the "CalibrationPath" within each configuration file to the correct location of the calibration files for the particular sensor
- Within each configuration set the "Bus" property of the IR_proxy_cam source to the correct USB bus number the sensor is connected to (Bus 0 means autodetect and should not be used when multiple sensors are connected)

Please Note: The number of simultaneously connected sensors is limited by the number of independent USB busses and the actual performance of the target machine.

Acquiring Images from erop-proxy-cam

Under /opt/eyerop/examples/ipccap there is the source code and binary library needed to compile the ipc-cap application provided with the SDK. In the folder you will find five files:

- ipccap.cpp – The main source file for the ipc-cap application where you have an example on how to use the SDK functions to acquire images
- op.cpp – Wrappers for low level objects provided for you by libop.a that already implement the shared memory access logic
- op.h – Header with object definitions needed to use the functionality provided by op.cpp
- libop.a – A statically linked library implementing all low-level access and synchronization to erop-proxy-cam
- libop.la – A libtool file for libop. It is recommended to link against libop.la and not directly agains libop.a

The IPC_tap is the "master" of the shared memory and synchronization objects used to hold the image data and manage the transfer. This means that when the video pipeline is stopped all shared objects will be deleted. Applications wishing to stop and start the video pipeline programmatically or load and unload the erop-proxy-cam daemon dynamically should release all open shared objects held by the client application and re-open them.

Synchronization of the client and IPC_tap is done using a shared single write multi read lock mechanism. This allows multiple clients to read from the same share memory. Client applications should make sure the properly release the read lock acquired on the shared memory once the memory is no longer needed. Holding a read lock on a buffer will prevent IPC_tap writing to it.

Using erop-tunnel to Control erop-proxy-cam and the Video Pipeline

Communication with the erop-proxy-cam daemon is done through a dedicated netlink channel and a propriety protocol. To facilitate the working with this mechanism the SDK provides erop-tunnel which is a small utility that receive a JSON script specifying the commands to send and outputs a JSON reply. The list of commands that can be sent to erop-proxy-cam can be found in the *Command Reference* section below.

The JSON script has the following syntax:

```
{
	"commands": [
		{
			"command_id": <String>,
			"target": <String>,
			"command_data": {
				<JSON of the command>
			}
		}
	]
}
```

"commands" is a JSON array of command objects. Commands are sent in sequential order

"command_id" is the id of the command (See the Command Reference section below)

"target" is the name of the erop-proxy-cam instance you want the command sent to. If this is omitted and not target is specified at the command line of erop-tunnel the command will be sent to all running instances

"command_data" is the JSON object of the command (See the Command Reference section below)

Here is an example of calling the GetVideoSourceProperties command:

```
{
"commands": [
	{
		"command_id": "GetVideoSourceProperties",
		"command_data": {
			"PropertyCategories": [
				"Camera Information",
				"Capture Parameters"
```

```
                                ]
                        }
                }
        ]
}
```

Erop-tunnel receives the -t /--target and -s/--script parameters at the command line

The -t option specifies the name of the target instance for the command. If the target name is omitted then the command will be sent to all the currently running instances unless the target is specified within the script

Ths -s option specifies the path to the JSON script to execute. If no script is specified then erop-tunnel expects its input from STDIN. If running in interactive mode you can type exit to quit the application.

Look at the python scripts under /opt/eyerop/examples/python to see how you can use erop-tunnel in your application

For a Hawkeye distribution you can find sample JSON scripts under /opt/eyerop/scripts/

Reference Manual

Command Reference

The erop-proxy-cam daemon exposes a JSON based message API that can be used to control and query the current running pipeline as well as to change the current pipeline.

ApplyActions

Please Note:
This command allows sending some action requests to the video pipeline. Although this command is enabled for all sensor types currently the specific actions listed in the table below are only available for the Hawkeye sensor.

REQUEST

Action	Syntax	Example

Request some action that changes the state of the running pipeline (General Syntax)	`{`
	`"Actions": [`
	`"Action1",`
	`…`
	`"ActionN"`
	`],`
	`"Action1": {`
	`"Param1_1": <Some value>`
	`…`
	`"ParamN_1": <Some value>`
	`},`
	`…`
	`"ActionN": {`
	`"Param1_N": <Some value>`
	`…`
	`"ParamN_N": <Some value>`
	`}`
	`}`
		`See specific examples below`

Initiate BIT		`{`
		`"Actions": [`
		`"DoBIT"`
		`],`
		`"DoBIT": {`
		`}`
		`}`

SetSyncExternal	`{`
	`"Actions": [`
	`"SetSyncExternal"`
	`],`
	`"SetSyncExternal": {`
	`"ExtSyncValue" : <Boolean>`
	`}`
	`}`
		`{`
		`"Actions": [`
		`"SetSyncExternal"`
		`],`
		`"SetSyncExternal": {`
		`"ExtSyncValue" : true`
		`}`
		`}`

ResetCBIT*	{
	"Actions": [
	"ResetCBIT"
	],
	"ResetCBIT": {
	"ResetCBITMask" : <HEX STR>
	}
	}
		{
		"Actions": [
		"ResetCBIT"
		],
		"ResetCBIT": {
		"ResetCBITMask" : "0xFF"
		}
		}

Set Pattern†	{
	"Actions": [
	"SetPattern"
	],
	"SetPattern": {
	"PatternId" : <Int [3,10]>
	}
	}
		{
		"Actions": [
		"SetPattern"
		],
		"SetPattern": {
		"PatternId" : 5
		}
		}

Update NUC		{
		"Actions": [
		"UpdateNUC"
		],
		"UpdateNUC": {
		}
		}

RESPONSE

Type	Syntax	Example
SUCCESS	{}	{}
ERROR	{}	{}

EnumVideoPipelines

REQUEST

Action	Syntax	Example

Get the list of supported pipelines	Empty JSON
	{
	}
		{
		}

RESPONSE

Type	Syntax	Example
SUCCESS	{ "ActivePipeline": "PipelineID", "CameraId": "camera-id", "PendingUpdates": <Boolean>, "PipelineRunning": <Boolean>, "SupportedPipelines": ["ID1",	{ "CameraId": "proxycam-ir", "ActivePipeline": "NV", "PipelineRunning": true, "PendingUpdates": false, "SupportedPipelines": ["NV",

* For the ResetCBITMask each test for which we wish to reset the error the corresponding bit should be set to ON. The reference for the CBIT mask can be found below in the reference for the *IR_proxy_cam* element

† To restart video streaming you my call the "SetSyncExternal" action with the desirable sync mechanism

	… "IDN"] }	"OPGAL EYE-Q™"] }
ERROR	{}	{}

GetVideoSourceProperties

REQUEST

Action	Syntax	Example

Get all properties of all the elements in the pipeline	Empty JSON object:
	{
	}
		{
		}

Get all the properties of a specified property category	{
	"PropertyCategories": [
	"Category1",
	…
	"CategoryN"
	]
	}
		Get the properties for categories "Image Processing" and "Camera Information"
		
		{
		"PropertyCategories": [
		"Image Processing",
		"Camera Information"
		]
		}

Get the value of specific properties	{
	"PropertyCategories": [
	"Category1",
	…
	"CategoryN"
	],
	"Category1": {
	"Properties": [
	"Property1",
	…
	"PropertyN"
	]
	},
	…
	"CategoryN": {
	"Properties": [
	"PropertyA",
	…
	"PropertyX"
	]
	}
	}
		Get the value of the "BlackHot" property
		
		{
		"PropertyCategories": [
		"Image Processing"
		],
		"Image Processing": {
		"Properties": [
		"BlackHot"
		]
		}
		}

RESPONSE

Type	Syntax	Example
SUCCESS	{ "ActivePipeline": "PipelineID", "CameraId": "camera-id", "PendingUpdates": <Boolean>,	Please note that the example below is truncated with most parts removed

	"PipelineRunning": <Boolean>,
	"VideoSourceConfiguration": {
	"PropertyCategories": [
	"Category1",
	…
	"CategoryN"
	]
	"Category1": {
	"Property1": {
	"Value": <value>
	},
	…
	"PropertyN": {
	"Value": <value>
	}
	},
	…
	"CategoryM": {
	"PropertyA": {
	"Value": <value>
	},
	…
	"PropertyX": {
	"Value": <value>
	}
	}
	}
	}
		{
		"ActivePipeline": "NV",
		"CameraId": "proxycam-ir",
		"PendingUpdates": false,
		"PipelineRunning": true,
		"VideoSourceConfiguration": {
		"PropertyCategories": [
		"Image Processing",
		"Camera Information",
		"Capture Parameters",
		"Internal"
		],
		"Image Processing": {
		"AtdfEnable": {
		"Value": true
		},
		"AtdfThreshold": {
		"Value": 350.0
		},
		"BlackHot": {
		"Value": false
		},
		"DrcGainLimit": {
		"Value": 350.0
		}
		}
		}

ERROR	{}	{}

SetVideoPipeline

Please Note:
"*SetVideoPipeline*" will not start the new pipeline. To start it you should call "*StartVideoPipeline*" upon successful return from this command

REQUEST

Action	Syntax	Example

Set the active video pipeline	{
	"PipelineId": "ID"
	}
		{
		"PipelineId": "NV"
		}

RESPONSE

Type	Syntax	Example

SUCCESS	Empty JSON
	{
	}
		{
		}

ERROR	{
	"ErrorMessage": "Message"
	}
		{
		"ErrorMessage": "Failed to set 'GGG' as active pipeline."
		}

SetVideoSourceProperties

Please Note:
After setting one or more properties one of the fields returned by the response is "PendingUpdates". If this field is true, then you should restart the pipeline by issuing "*StopVideoPipeline*" followed by "*StartVideoPipeline*" for all the changes you made to take effect.

REQUEST

Action	Syntax	Example

Set the value of one or more properties	{
	"VideoSourceConfiguration": {
	"Category1": {
	"Property1": <value>,
	…
	"PropertyN": <value>
	},
	…
	"CategoryM": {
	"PropertyA": <value>,
	…
	"PropertyX": <value>
	}
	}
	}
		Set the "BlackHot" property to true
		
		{
		"VideoSourceConfiguration": {
		"Image Processing": {
		"BlackHot": true
		}
		}
		}

RESPONSE

Type	Syntax	Example
SUCCESS	Same as GetVideoSourceProperties	Same as GetVideoSourceProperties
ERROR	{}	{}

StartVideoPipeline

REQUEST

Action	Syntax	Example

Start the active video pipeline	{
	"ApplyPendingUpdates": <Boolean>
	}
	
	Please Note:
	Although "ApplyPendingUpdates" is optional and it is possible to pass false to this property it is recommended that this property is always specified with a value of true. Failing to do so can cause the changes to some parameters not to take effect
		{
		"ApplyPendingUpdates": true
		}

RESPONSE

Type	Syntax	Example

SUCCESS	{
	"CameraId":"camera id",
	"ActivePipeline":"ID",
	"PipelineRunning": <Boolean>,
	"PendingUpdates": <Boolean>
	}
		{
		"CameraId":"proxycam-ir",
		"ActivePipeline":"OPGAL EYE-Q™",
		"PipelineRunning": true,
		"PendingUpdates": false
		}

ERROR	{
	"ErrorMessage": "Message"
	}
		{
		}

StopVideoPipeline

REQUEST

Action	Syntax	Example
Stop the active video pipeline		{ }

RESPONSE

Type	Syntax	Example

SUCCESS	{
	"CameraId":"camera id
	}
		{
		"CameraId":"proxycam-ir
		}

ERROR	{
	"ErrorMessage": "Message"
	}
		{
		}

Opgal Optronic Industries Ltd.
www.opgal.com
Tel: +972 4 995-3903
Fax: +972 4 995-3900
PO Box 462, Industrial Area 5, Karmiel 2161401, Israel

Configuration Reference

This section is incomplete

appcomands.conf

This section is incomplete

erop.conf

This section is incomplete

proxycam-ir.conf

This section is incomplete

routing.conf

This section is incomplete

sources.conf

This section is incomplete

users.conf

This section is incomplete

Opgal Optronic Industries Ltd.
www.opgal.com
Tel: +972 4 995-3903
Fax: +972 4 995-3900
PO Box 462, Industrial Area 5, Karmiel 2161401, Israel

Video Pipeline Elements Reference

Please Note: Not all possible configuration properties are listed and some of the properties listed are left undocumented.

Property Category

Each property belongs to a property category. The category is listed in the **category** column of the property table.

It is recommended not to change the values of properties outside the "Image Processing" category.

Access Control

The SDK supports a sophisticated user-based access control mechanism at the property level. However, this mechanism is not documented within this document.

This document assumes that all access is done using a privileged "Developer" user. Never the less some properties are read only, and some properties cannot be modified at runtime. These restrictions are listed in the **Access** column of the property table.

Access columns takes one of the following values:

- RO – Read Only
- RW – Read Write
- WOC – Write Only through Configuration file

Common Element Properties

All elements supplied by the SDK support the following properties

Property	Description
"AdvertiseCommonProps"	**Type**: BOOL **Access**: WOC **Category**: Internal **Default**: true This property can only be set from the configuration file. If this property is set to false, then the properties listed in this table will not be available for querying or manipulating at runtime. All properties can be set from the configuration file regardless if the value of this property
"AdvertiseProps"	**Type**: BOOL **Access**: WOC **Category**: Internal **Default**: true This property can only be set from the configuration file. If this property is set to false, then no properties of the element will be available for querying or manipulating at runtime. All properties can be set from the configuration file regardless of the value of this property
"BuildDate"	**Type**: STR **Access**: RO **Category**: Internal **Default**: - The build date of the plugin library
"CountIncomingFrames"	**Type**: BOOL **Access**: RW **Category**: Internal **Default**: false This property has no effect when using the SDK. It is better to leave it at false
"LogIncomingFrames"	**Type**: BOOL **Access**: RW **Category**: Internal **Default**: false

	This property can be set to true to log a line with the frame id and the frame counter for each valid incoming frame passing through the element
"PluginVersion"	**Type**: STR **Access**: RO **Category**: Internal **Default**: - The version of the plugin library
"Verbosity Level"	**Type**: INT **Access**: RW **Category**: Internal **Default**: 1 The property can be used to increase the verbosity of the plugins output to log and console. Range [0-2]

Hdr_16

FILTER element to perform Dynamic Range Compression which takes 16bit dynamic range and compress it to 8bit dynamic range with High Dynamic Range enhancement. This filter is our most sophisticated DRC to date and has automatic design parameters adjustment according to the scene as well as automatic spatial noise reduction and automatic image enhancement.

Please Note: This filter is designed to work directly with the data output of the NUC data after the NORMALIZER. Do not use ***TA_nv_hpf_16*** before this element – It will hurt both performance and image quality.

INPUT	GRAY16 Signed
OUTPUT	GRAY16 Unsigned (only lower byte valid) or YUV420. Output is automatically determined by the requirements of the next element in the pipeline

Algorithm Description

This algorithm bundles several algorithms into one controlled self-adjusting set of algorithms in the following order:

LOWPASS

f_V3(V3_Strength, AutoAdjust(V3 Alpha))

EYEQ

f_drc(HDR(),DRC())

EDGE ENHANCEMENT

Exposes the following configuration properties

Property	Description
"Auto LPF Threshold"	**Type**: DBL **Access**: RW **Category**: Image Processing **Default**: 5.0 If you have relatively high energy scenes and you experience that the LOWPASS is too strong you may want to decrease this number. Range: [1.0,20.0]

"BlackHot"	**Type**: BOOL **Access**: RW **Category**: Image Processing **Default**: false If true, the output image will show hot areas as dark and cold areas as light
"DrcGainLimit"	**Type**: DBL **Access**: RW **Category**: Image Processing **Default**: 200.0 Set the lower bound limit for number of active image channels considered for DRC. ActiveChannels = MIN ("DrcGainLimit", ActiveChannels). This in effect will limit the amount of stretching done for low channel count images. Range: [0.0,2000.0]
"Edge Contrast"	**Type**: INT **Access**: RW **Category**: Image Processing **Default**: 120 Set the strength of the edge enhancement. Setting this value to 0 is akin to disabling the edge enhancement Range: [0,1000]
"Edge Enhancement"	**Type**: BOOL **Access**: RW **Category**: Image Processing **Default**: true Enable/Disable EDGE ENHANCMENT
"Edge Limit"	**Type**: DBL **Access**: RW **Category**: Image Processing **Default**: 24.0 When EDGE ENHANCEMENT is enabled it is sometimes desirable to limit the emphasis of some edges. [0.0,128.0]
"Gamma"	**Type**: DBL **Access**: RW **Category**: Image Processing **Default**: 1.0 Gamma Correction Range: [0.5,3.0]
"Gray Levels"	**Type**: INT **Access**: RW **Category**: Image Processing **Default**: 200 Limit the number of output gray level from HDR. It is desirable not to allow HDR to utilize the entire 8bit range so that some gray levels be left for the EDGE ENHANCEMENT algorithm. When the output is directed to some hardware unable to display or process 8bits of data it may be even desirable to reduce this number even more. Range: [100,255]
"Hdr Threshold"	**Type**: DBL **Access**: RW **Category**: Image Processing **Default**: 2000.0 Design parameter that defines the sensitivity of the HDR to the number of active channels in the image Range: [1000.0,6000.0]
"HdrFactor"	**Type**: DBL **Access**: RW **Category**: Image Processing **Default**: 18.0 The HDR algorithm tries to find a suitable representation to all active channels in the image and avoid change in gray level output when

	the scene changes. This makes the output of the HDR more suitable to analytics engines. However, many times the output of the pure HDR image is not pleasant to the human eye. This parameter adjusts the "pleasantness" of the image. The higher the number the more pleasant the image but less dynamic range is considered. [1.0,20.0]
"HistogramBits"	**Type**: INT **Access**: RW **Category**: Image Processing **Default**: 16 Do not change this value
"HistogramCalcInterval"	**Type**: INT **Access**: RW **Category**: Image Processing **Default**: 4 For most scenes it is not necessary and even damaging to calculate a new histogram every frame. The histogram will be calculated every "HistogramCalcInterval" frames. Possible values are the following powers of two: 1|2|4|8|16|32|64|128|256|512
"HistogramTrim"	**Type**: STR **Access**: RW **Category**: Image Processing **Default**: "0.1%" During the life of a thermal product new bad pixels not existing at production time appear on the sensor. These bad pixels generally produce signals at the edges of the histogram. This parameter is used to trim the edges of the histogram and eliminate them from the Active Channel Count calculation. It is recommended not to change this value. Possible values are: "NO TRIM" | "0.1%" | "0.5%" | "1%" | "2.5%" | "5%" | "10%"
"HistStats"	**Type**: STR **Access**: RO **Category**: Camera Information **Default**: - Reports the calculated histogram statistics in the form:
"ImageNoiseLevelPixelCountThreshold"	**Type**: INT **Access**: RW **Category**: Image Processing **Default**: 0 Set the number of pixels in a histogram bin required for the bin to be considered as an active channel. A value of 0 means that this number is automatically calculated depending on the histogram shape and image. It is best to leave this number at 0. Range: [0,1000]
"LutHistoryAlpha"	**Type**: INT **Access**: RW **Category**: Image Processing **Default**: 5 For most scenes it is desirable not to have the scene histogram output to change when a passing cold or hot object passes through the frame. This parameter determines the strength of the historical histogram compared to the current histogram. The higher this number is the slower changes will affect the scene. For stable scenes values of 4 or 5 are recommended. A value of 0 will disable the history. Use this property together with "HistogramCalcInterval" to determine the speed of scene adjustment. Range: [0,11]

"V3 Alpha"	**Type**: INT **Access**: RW **Category**: Image Processing **Default**: 70.0 The "V3 Alpha" is automatically adjusted internally between: "V3 Alpha" – 20.0 and 92.0 according to the scene Range: [0.0,92.0] **Note:** Value range may change in the future
"V3 Strength"	**Type**: INT **Access**: RW **Category**: Image Processing **Default**: 8 Design parameter for the f_V3 LOWPASS function. The higher this number the greater the effect of the function [0,12] **Note:** Value range may change in the future

Image_normalizer_16

FILTER element used to normalize NUC results to a value range that is understood by ***TA_nv_drc_16*** and ***Hdr_16*** filters

INPUT	GRAY16 Signed
OUTPUT	GRAY16 Signed

This element exposes no properties

IPC_tap

TAP element used to copy frames into shared memory that can be accessed by a 3rd party application

INPUT	ANY
OUTPUT	SAME AS INPUT

Exposes the following configuration properties

Property	Description
"IPC_Log2_Buffers"	**Type**: INT **Access**: WOC **Category**: Internal **Default**: 3 The log2 number of shared memory buffers allocated by the element. The number of buffers allocated is (1<< IPC_Log2_Buffers). Applications interacting with the element should allocate the same number of buffers as they are filled in a cyclic manner.
"IPC_Buffer_Size"	**Type:** INT **Access:** WOC **Category:** Internal **Default:** 640x480x2 + 64
"IPC_Copy_Header"	**Type:** BOOL **Access:** RW **Category:** Internal **Default:** true

IR_proxy_cam

SOURCE element for working against Opgal USB devices.

INPUT	-
OUTPUT	GRAY16 Unsigned

Exposes the following configuration properties

Opgal Optronic Industries Ltd.
www.opgal.com
Tel: +972 4 995-3903
Fax: +972 4 995-3900
PO Box 462, Industrial Area 5, Karmiel 2161401, Israel

Property	Description
"Bus"	**Type**: INT **Access**: WOC **Category**: Internal **Default**: 0 The USB Bus the sensor is connected to. Supported range is [0,127]. Use 0 for auto detection. If you have more than one sensor connected to the system you should make sure to connect each one to a different bus and to specifically set the correct bus number in the configuration file
"CaptureMode"	**Type**: STR **Access**: RW **Category**: Capture Parameters **Default**: - A string representing the capture mode used by the sensor. The sensor reports the supported capture modes. It is possible for the application to select a capture mode from that list. The capture mode string takes the form of: "GRAY16 [c0,r0,cols,rows] @ fps HZ" c0 – ROI column r0 – ROI row cols – ROI columns rows – ROI rows fps – The capture FPS
"CommandOpCode"	**Type**: INT **Access**: RO **Category**: Camera Information **Default**: -
"Dfpa"	**Type**: INT **Access**: RO **Category**: Camera Information **Default**: -
"FirmwareVersion"	**Type**: STR **Access**: RO **Category**: Camera Information **Default**: - The firmware number of the sensor
"GAIN"	**Type**: INT **Access**: RO **Category**: Camera Information **Default**: -
"HardwareVersion"	**Type**: INT **Access**: RO **Category**: Camera Information **Default**: - The hardware version of the sensor
"ProductId"	**Type**: STR(HEX) **Access**: WOC **Category**: Internal **Default**: "0x0002" Sets the Product Id for the supported USB based sensor connected to the system. At this time all sensors supported by the SDK use the same Product Id so do not change the default
"SensorCaptureFrequency"	**Type**: DBL **Access**: RO **Category**: Camera Information **Default**: - Reports the capture FPS from the sensor divided by 100. Please note that this is not the actual FPS the pipeline is working at. Also, the interpretation of this value varies across sensors. To get the FPS the pipeline is operating at please See the command reference for the GetVideoSourceProperties command
"SerialNumber"	**Type**: INT **Access**: RO **Category**: Camera Information **Default**: - The serial number for the sensor
"SoftwareVersion"	**Type**: STR **Access**: RO **Category**: Camera Information **Default**: - The onboard embedded software version of the sensor if applicable
"StartCode"	**Type**: INT **Access**: WOC **Category**: Internal **Default**: 2 This value should not be changed
"Tfpa"	**Type**: DBL **Access**: RO **Category**: Camera Information **Default**: -

	Reports the temperature of the sensor as measured at the sensor
"TINT"	**Type**: INT **Access**: RO **Category**: Camera Information **Default**: -
"UsbPowerControlGpioUri"	**Type**: STR **Access**: WOC **Category**: Internal **Default**: - If the system has a GPIO pin mappable through sysfs that can be used to control the power to the USB connected to the sensor then this property can be set to point to this pin. If the GPIO pin is successfully mapped, then the source element will power cycle the device at startup. The format of the GPIO URI is as follows: sirq://gpio?pin=<pin#>&type=Out Other optional URI parameters: unexport_on_exit=True|False (Default is False) activelow=True|False (Default is False)
"UsbPowerDownTime"	**Type**: INT **Access**: WOC **Category**: Internal **Default**: 2750 The time in milliseconds to wait before powering up the device after power down. This value has an effect only if a valid power control GPIO is set. Valid range for this property is [500,10000]
"UsbTransferSize"	**Type**: INT **Access**: RO **Category**: Internal **Default**: 8192 Do not change this value
"VendorId"	**Type**: STR(HEX) **Access**: WOC **Category**: Internal **Default**: "0x1772" Sets the Vendor Id for the supported USB based sensor connected to the system. At this time all sensors supported by the SDK use the same Vendor Id so do not change the default
Therm-App® and Therm-App® PRO properties	
"Tcase"	**Type**: INT **Access**: RO **Category**: Camera Information **Default**: -
"VBus"	**Type**: INT **Access**: RO **Category**: Camera Information **Default**: -
"Vdda"	**Type**: INT **Access**: RO **Category**: Camera Information **Default**: -
"Vgfid"	**Type**: INT **Access**: RO **Category**: Camera Information **Default**: -
"Vgsk"	**Type**: INT **Access**: RO **Category**: Camera Information **Default**:
"Vsk"	**Type**: INT **Access**: RO **Category**: Camera Information **Default**: -
"Vtbias"	**Type**: INT **Access**: RO **Category**: Camera Information **Default**: -
"Vtofs1"	**Type**: INT **Access**: RO **Category**: Camera Information **Default**: -
"Vtofs2"	**Type**: INT **Access**: RO **Category**: Camera Information **Default**: -
Hawkeye properties	
CBIT_result	**Type**: STR(HEX) **Access**: RO **Category**: Internal **Default**: "0x0" Reports the continuous BIT results. A bit is assigned for each test. An ON status for a bit indicates an error. CBIT is mapped as following:

BIT	TEST
0 (LSB)	Flash
1	N/A
2	N/A
3	Detector

<table>
<tr><td></td><td>
<table>
<tr><td>4</td><td>Temperature</td></tr>
<tr><td>5</td><td>Power</td></tr>
<tr><td>6</td><td>UART</td></tr>
<tr><td>7 (MSB)</td><td>External Sync</td></tr>
</table>
</td></tr>
<tr><td>ExtSyncEnable</td><td>Type: BOOL Access: RO Category: Camera Information Default: false</td></tr>
<tr><td>IBIT_result</td><td>Type: STR(HEX) Access: RO Category: Internal Default: "0x0"

Reports the initiated BIT results which check on board calibration tables. A bit is assigned for each table. An ON status for a bit indicates an error.
IBIT is mapped as following:
<table>
<tr><th>BIT</th><th>TEST</th></tr>
<tr><td>0 (LSB)</td><td>TBL_0</td></tr>
<tr><td>1</td><td>TBL_1</td></tr>
<tr><td>2</td><td>TBL_2</td></tr>
<tr><td>3</td><td>TBL_3</td></tr>
<tr><td>4</td><td>TBL_4</td></tr>
<tr><td>5</td><td>TBL_5</td></tr>
<tr><td>6</td><td>TBL_6</td></tr>
<tr><td>7</td><td>TBL_7</td></tr>
<tr><td>8</td><td>TBL_8</td></tr>
<tr><td>9</td><td>TBL_9</td></tr>
<tr><td>10</td><td>TBL_10</td></tr>
<tr><td>11</td><td>TBL_11</td></tr>
<tr><td>12</td><td>TBL_12</td></tr>
<tr><td>13</td><td>TBL_13</td></tr>
<tr><td>14</td><td>TBL_14</td></tr>
<tr><td>15</td><td>TBL_15</td></tr>
<tr><td>16</td><td>TBL_16</td></tr>
<tr><td>17</td><td>TBL_17</td></tr>
<tr><td>18</td><td>TBL_18</td></tr>
<tr><td>19 (MSB)</td><td>TBL_19</td></tr>
</table>
</td></tr>
<tr><td>IBitInProgress</td><td>Type: BOOL Access: RO Category: Camera Information Default: false</td></tr>
<tr><td>PBIT_result</td><td>Type: STR(HEX) Access: RO Category: Internal Default: "0x0"

Reports the Power Up BIT results. A bit is assigned for each test. An ON status for a bit indicates an error.
PBIT is mapped as following:
<table>
<tr><th>BIT</th><th>TEST</th></tr>
<tr><td>0 (LSB)</td><td>Flash</td></tr>
<tr><td>1</td><td>N/A</td></tr>
<tr><td>2</td><td>N/A</td></tr>
<tr><td>3</td><td>Detector</td></tr>
<tr><td>4</td><td>IIC</td></tr>
<tr><td>5</td><td>N/A</td></tr>
<tr><td>6</td><td>Application Configuration</td></tr>
<tr><td>7</td><td>RRB</td></tr>
<tr><td>8</td><td>N/A</td></tr>
</table>
</td></tr>
</table>

	9 N/A 10 Detector Configuration 11 (MSB) CDS
NumOfRanges	**Type**: INT **Access**: RO **Category**: Camera Information **Default**: -
Rst0	**Type**: INT **Access**: RO **Category**: Camera Information **Default**: -
Trange	**Type**: INT **Access**: RO **Category**: Camera Information **Default**: -

Null_sink

SINK element that discards incoming input

INPUT	ANY
OUTPUT	-

Exposes the following configuration properties

Property	Description
"ForceImageFormat"	**Type**: STR **Access**: WOC **Category**: Image Processing **Default**: "" If this property is set, then the Null_sink will report upstream that it requires input of the specified type. Possible values: "GRAY16" | "YUV420" | "ANY" | ""

NV_atdf_16

FILTER element with that removes temporal noise

INPUT	GRAY16 Unsigned
OUTPUT	GRAY16 Unsigned

Exposes the following configuration properties

Property	Description
"AtdfEnable"	**Type**: BOOL **Access**: RW **Category**: Image Processing **Default**: true Used to enable or disable the operation of the filter
"AtdfThreshold"	**Type**: DBL **Access**: RW **Category**: Image Processing **Default**: 350.0 Used to set the strength of the noise removal. The higher the setting the stronger the filter will eliminate temporal noise. Please note that at higher settings some smearing of the image on movement may be visible. Accepted value range [64.0,512.0] **Please Note:** The value range for this property may change in the future

Screen_sink

SINK element that writes the output to /dev/fbN

INPUT	UYVY, VYUY, YUYV, YVYU, YUV420, BGRA32, RGB565 Better pass the native frame buffer format
OUTPUT	-

Exposes the following configuration properties

Property	Description
"FramebufferName"	**Type**: STR **Access**: WOC **Category**: VideoOut **Default**: "fb0" The name of the frame buffer to use
"OutputPattern"	**Type**: STR **Access**: RW **Category**: VideoOut **Default**: "None" "None" | "WRGBK"
"AutoPowerOn"	**Type**: BOOL **Access**: RW **Category**: VideoOut **Default**: false Should we unblank the device at startup
"VideoOutMode"	**Type**: STR **Access**: RW **Category**: VideoOut **Default**: - Get or set the capture video mode. Possible values depend on the actual frame buffer
"DisableConsole"	**Type**: BOOL **Access**: RW **Category**: Internal **Default**: true Should we enter graphics mode

SiiOP_Adr

CAPTURE FILTER element to maintain the correct image level output from the sensor

INPUT	GRAY16 Unsigned
OUTPUT	-

Exposes the following configuration properties

Property	Description
"DgskHistScale"	**Type**: INT **Access**: RW **Category**: Internal **Default**: 12 Do not change this value
"DgskImageBits"	**Type**: INT **Access**: RW **Category**: Internal **Default**: - Do not change this value
"DgskSlope"	**Type**: DBL **Access**: RW **Category**: Internal **Default**: 0.01 Do not change this value
"StartAdrOff"	**Type**: BOOL **Access**: RW **Category**: Internal **Default**: - Do not change this value

TA_nv_drc_16

FILTER element to perform Dynamic Range Compression which takes 16bit dynamic range and compress it to 8bit dynamic range

INPUT	GRAY16 Signed
OUTPUT	GRAY16 Unsigned (only lower byte valid) or YUV420. Output is automatically determined by the requirements of the next element in the pipeline

Exposes the following configuration properties

Property	Description
"BlackHot"	**Type**: BOOL **Access**: RW **Category**: Image Processing **Default**: false If true, the output image will show hot areas as dark and cold areas as light
"DrcGainLimit"	**Type**: DBL **Access**: RW **Category**: Image Processing **Default**: 350.0 Set the lower bound limit for number of active image channels considered for DRC. ActiveChannels = MIN ("DrcGainLimit", ActiveChannels). This in effect will limit the amount of stretching done for low channel count images. Range: [0.0,2000.0]
"HistogramBits"	**Type**: INT **Access**: RW **Category**: Image Processing **Default**: 16 Do not change this value
"HistogramCalcInterval"	**Type**: INT **Access**: RW **Category**: Image Processing **Default**: 4 For most scenes it is not necessary and even damaging to calculate a new histogram every frame. The histogram will be calculated every "HistogramCalcInterval" frames. Possible values are the following powers of two: 1|2|4|8|16|32|64|128|256|512
"HistogramTrim"	**Type**: STR **Access**: RW **Category**: Image Processing **Default**: "0.1%" During the life of a thermal product new bad pixels not existing at production time appear on the sensor. These bad pixels generally produce signals at the edges of the histogram. This parameter is used to trim the edges of the histogram and eliminate them from the Active Channel Count calculation. It is recommended not to change this value. Possible values are: "NO TRIM" | "0.1%" | "0.5%" | "1%" | "2.5%" | "5%" | "10%"
"HistStats"	**Type**: STR **Access**: RO **Category**: Image Processing **Default**: - Reports the calculated histogram statistics in the form:
"ImageNoiseLevelPixelCountThreshold"	**Type**: INT **Access**: RW **Category**: Image Processing **Default**: 0

	Set the number of pixels in a histogram bin required for the bin to be considered as an active channel. A value of 0 means that this number is automatically calculated depending on the histogram shape and image. It is best to leave this number at 0. Range: [0,1000]
"LutHistoryAlpha"	**Type**: INT **Access**: RW **Category**: Image Processing **Default**: 5 For most scenes it is desirable not to have the scene histogram output to change when a passing cold or hot object passes through the frame. This parameter determines the strength of the historical histogram compared to the current histogram. The higher this number is the slower changes will affect the scene. For stable scenes values of 4 or 5 are recommended. A value of 0 will disable the history. Use this property together with "HistogramCalcInterval" to determine the speed of scene adjustment. Range: [0,11]

TA_nv_hpf_16

FILTER element that can perform several image enhancement algorithms

INPUT	GRAY16 Signed
OUTPUT	GRAY16 Signed

Algorithm Description

This filter offers a selection between 4 modes of operation:

NO CHANGE:

BASIC: OUT = IN

HIGHPASS

IE1: OUT = IN – V1_HpfLapha * f_V1(V1_LpfAlpha)

IE2: OUT = IN – V2_HpfAlpha * f_V2(V2_HpfThreshold, V2_LpfAlpha)

LOWPASS

IE3: OUT = f_V3(V3_Strength, V3 Alpha)

Exposes the following configuration properties

Property	Description
"Image Enhancement Algorithm"	**Type**: STR **Access**: RW **Category**: Image Processing **Default**: "IE 3" Possible values: "BASIC"

	"IE 1" "IE 2" "IE 3"
"V1 HpfAlpha"	**Type**: DBL **Access**: RW **Category**: Image Processing **Default**: 1.0 Increase or decrease the weight of the LPF component of the HPF function. If set to 0 then OUT = IN Range: [0.0,1.0]
"V1 LpfAlpha"	**Type**: DBL **Access**: RW **Category**: Image Processing **Default**: 0.94 Design parameter for the f_V1 LPF function. The higher this number the greater the effect of the function [0.0,1.0]
"V2 HpfAlpha"	**Type**: DBL **Access**: RW **Category**: Image Processing **Default**: 0.9 Increase or decrease the weight of the LPF component of the HPF function. If set to 0 then OUT = IN Range: [0.0,1.0]
"V2 HpfThreshold"	**Type**: DBL **Access**: RW **Category**: Image Processing **Default**: 256.0 Design parameter for the f_V2 LPF function. The higher this number the greater the effect of the function [0.0,4096.0] **Note:** Value range may change in the future
"V2 LpfAlpha"	**Type**: DBL **Access**: RW **Category**: Image Processing **Default**: 0.8 Design parameter for the f_V2 LPF function. The higher this number the greater the effect of the function [0.0,1.0]
"V3 Alpha"	**Type**: DBL **Access**: RW **Category**: Image Processing **Default**: 80.0 Design parameter for the f_V3 LPF function. The higher this number the greater the effect of the function [0.0.92.0] **Note:** Value range may change in the future
"V3 Strength"	**Type**: INT **Access**: RW **Category**: Image Processing **Default**: 8 Design parameter for the f_V3 LPF function. The higher this number the greater the effect of the function [0,12] **Note:** Value range may change in the future

TA_nv_nuc_bpr_16

FILTER element that performs non-uniformity correction and bad-pixel removal. The NUC algorithm in use will be automatically selected by examining the calibration tables.

INPUT	GRAY16 Unsigned
OUTPUT	GRAY16 Signed

Opgal Optronic Industries Ltd.
www.opgal.com
Tel: +972 4 995-3903
Fax: +972 4 995-3900
PO Box 462, Industrial Area 5, Karmiel 2161401, Israel

Exposes the following configuration properties

Property	Description
"CalibType"	**Type**: INT **Access**: RW **Category**: Internal **Default**: 65535 This property can be used to override the automatically discovered NUC algorithm. Do not set this value unless directed to do so by Opgal's customer support
"NoiseBitCount"	**Type**: INT **Access**: RW **Category**: Image Processing **Default**: 2 During some types of NUC some extra noise can be removed. This value gives an estimate as to the noise level in bits. For most NUC algorithms this value has no effect and for those it does, it is not recommended to change this value. Valid range [0,5]
"PostNucCorrection"	**Type**: BOOL **Access**: RW **Category**: Image Processing **Default**: true Some NUC algorithm have an optional post NUC step which can be separately enabled or disabled. For most NUC algorithms this value has no effect and for those it does, it is not recommended to change this value.

Frame Header Reference

The header for the frame is 32 16bit words long as specified below (For readability documented offsets are present in the table). The header is received at the beginning of every frame unless the "IPC_Copy_Header" property of the *IPC_tap* element is set to false.

Offset	Field	Hawkeye	ThermApp-pro	ThermApp	Comment
0	PREAMBLE0	A5A5	A5A5	A5A5	
1	PREAMBLE1	A5A5	A5A5	A5A5	
2	PREAMBLE2	A5A5	A5A5	A5A5	
3	PREAMBLE3	A5D5	A5D5	A5D5	
4	OPCODE	2	2	2	This is the value for regular video. Other values may be possible
5	S/N LSB	-	-	-	The 32bit serial number of the detector LSB
6	S/N MSB	-	-	-	The 32bit serial number of the detector MSB
9	ROI TOP	0	0	120	For Therm-App® PRO it may be possible to get other values other than 0 here.
10	ROI LEFT	0	0	180	For Therm-App® PRO it may be possible to get other values other than 0 here
11	ROWS	1E0	1E0	120	For Therm-App® PRO it may be possible to get other values other than 1E0 here.
12	COLS	280	280	180	For Therm-App® PRO it may be possible to get other values other than 280 here.

Opgal Optronic Industries Ltd.
www.opgal.com
Tel: +972 4 995-3903
Fax: +972 4 995-3900
PO Box 462, Industrial Area 5, Karmiel 2161401, Israel

13	FPS	2500	{2500,870}	{25,9}	For Hawkeye and Therm-App® PRO the frame rate is multiplied by 100
20	EXT SYNC IND	{1,0}	N/A	N/A	For Hawkeye Only
24	IBIT IN PROGRESS	{1,0}	N/A	N/A	For Hawkeye Only
26	FRAME ID LSB	-	-	-	32bit Frame number LSB
27	FRAME ID MSB	-	-	-	32bit Frame number MSB
31	DATA FORMAT	2	1	{0,FFFF}	Can be used to identify the attached sensor