Support

Downloads

Knowledge Base

Contact Support

xiAPI Manual

Table of Contents

XI_PRM_LUT_EN or “LUTEnable”
XI_PRM_LUT_INDEX or “LUTIndex”
XI_PRM_LUT_VALUE or “LUTValue”
XI_PRM_TRG_SOURCE or “trigger_source”
XI_PRM_TRG_SOFTWARE or “trigger_software”
XI_PRM_TRG_DELAY or "trigger_delay"
XI_PRM_AVAILABLE_BANDWIDTH "available_bandwidth"
XI_PRM_LIMIT_BANDWIDTH "limit_bandwidth"
XI_PRM_LIMIT_BANDWIDTH_MODE "limit_bandwidth_mode"
XI_PRM_SENSOR_TAPS "sensor_taps"
XI_PRM_SENSOR_CLOCK_FREQ_HZ "sensor_clock_freq_hz"
XI_PRM_SENSOR_CLOCK_FREQ_INDEX "sensor_clock_freq_index"
XI_PRM_SENSOR_OUTPUT_CHANNEL_COUNT "sensor_output_channel_count"
XI_PRM_SENSOR_DATA_BIT_DEPTH "sensor_bit_depth"
XI_PRM_OUTPUT_DATA_BIT_DEPTH "output_bit_depth"
XI_PRM_IMAGE_DATA_BIT_DEPTH "image_data_bit_depth"
XI_PRM_OUTPUT_DATA_PACKING "output_bit_packing"
XI_PRM_OUTPUT_DATA_PACKING_TYPE "output_bit_packing_type"
XI_PRM_ACQ_TIMING_MODE "acq_timing_mode"
XI_PRM_TRG_SELECTOR "trigger_selector"
XI_PRM_ACQ_FRAME_BURST_COUNT "acq_frame_burst_count"
XI_PRM_IMAGE_USER_DATA "image_user_data"

Color management settings

XI_PRM_WB_KR or “wb_kr”
XI_PRM_WB_KG or “wb_kg”
XI_PRM_WB_KB or “wb_kb”
XI_PRM_MANUAL_WB or “manual_wb”
XI_PRM_AUTO_WB or “auto_wb”
XI_PRM_GAMMAY or “gammaY”
XI_PRM_GAMMAC or “gammaC”
XI_PRM_SHARPNESS or “sharpness”
XI_PRM_CC_MATRIX_00 or “ccMTX00”
XI_PRM_CC_MATRIX_01 or “ccMTX01”
XI_PRM_CC_MATRIX_02 or “ccMTX02”
XI_PRM_CC_MATRIX_03 or “ccMTX03”
XI_PRM_CC_MATRIX_10 or “ccMTX10”
XI_PRM_CC_MATRIX_11 or “ccMTX11”
XI_PRM_CC_MATRIX_12 or “ccMTX12”
XI_PRM_CC_MATRIX_13 or “ccMTX13”
XI_PRM_CC_MATRIX_20 or “ccMTX20”
XI_PRM_CC_MATRIX_21 or “ccMTX21”
XI_PRM_CC_MATRIX_22 or “ccMTX22”
XI_PRM_CC_MATRIX_23 or “ccMTX23”
XI_PRM_CC_MATRIX_30 or “ccMTX30”
XI_PRM_CC_MATRIX_31 or “ccMTX31”
XI_PRM_CC_MATRIX_32 or “ccMTX32”
XI_PRM_CC_MATRIX_33 or “ccMTX33”
XI_PRM_DEFAULT_CC_MATRIX or “defccMTX”
XI_PRM_IMAGE_IS_COLOR or “iscolor”
XI_PRM_COLOR_FILTER_ARRAY or "cfa"
XI_PRM_CMS or "cms"
XI_PRM_CMS_INTENT or "cms_intent"
XI_PRM_APPLY_CMS or "apply_cms"
XI_PRM_INPUT_CMS_PROFILE or "input_cms_profile"
XI_PRM_OUTPUT_CMS_PROFILE or "output_cms_profile"

Device IO parameters.

XI_PRM_GPI_SELECTOR or “gpi_selector”
XI_PRM_GPI_MODE or “gpi_mode”
XI_PRM_GPI_LEVEL or “gpi_level”
XI_PRM_GPO_SELECTOR or “gpo_selector”
XI_PRM_GPO_MODE or “gpo_mode”
XI_PRM_LED_SELECTOR or “led_selector”
XI_PRM_LED_MODE or “led_mode”
XI_PRM_TS_RST_SOURCE “ts_rst_source”
XI_PRM_TS_RST_MODE “ts_rst_mode”

Extended Device Parameters.

XI_PRM_ACQ_BUFFER_SIZE or “acq_buffer_size”
XI_PRM_ACQ_BUFFER_SIZE_UNIT or "acq_buffer_size_unit"
XI_PRM_ACQ_TRANSPORT_BUFFER_SIZE or "acq_transport_buffer_size"
XI_PRM_ACQ_TRANSPORT_PACKET_SIZE or "acq_transport_packet_size"
XI_PRM_BUFFERS_QUEUE_SIZE or “buffers_queue_size”
XI_PRM_ACQ_TRANSPORT_BUFFER_COMMIT or “acq_transport_buffer_commit”
XI_PRM_RECENT_FRAME or “recent_frame”
XI_PRM_TRANSPORT_DATA_TARGET or “transport_data_target”
XI_PRM_DEVICE_RESET or “device_reset”
XI_PRM_AEAG or “aeag”
XI_PRM_AE_MAX_LIMIT or “ae_max_limit”
XI_PRM_AG_MAX_LIMIT or “ag_max_limit”
XI_PRM_EXP_PRIORITY or “exp_priority”
XI_PRM_AEAG_LEVEL or “aeag_level”
XI_PRM_AEAG_ROI_OFFSET_X or “aeag_roi_offset_x”
XI_PRM_AEAG_ROI_OFFSET_Y or “aeag_roi_offset_y”
XI_PRM_AEAG_ROI_WIDTH or “aeag_roi_width”
XI_PRM_AEAG_ROI_HEIGHT or “aeag_roi_height”
XI_PRM_DEBOUNCE_EN or “dbnc_en”
XI_PRM_DEBOUNCE_T0 or “dbnc_t0”
XI_PRM_DEBOUNCE_T1 or “dbnc_t1”
XI_PRM_DEBOUNCE_POL or “dbnc_pol”

Temperature

XI_PRM_IS_COOLED or "iscooled"
XI_PRM_COOLING or "cooling"
XI_PRM_TARGET_TEMP or "target_temp"
XI_PRM_TEMP_SELECTOR or "temp_selector"
XI_PRM_TEMP or "temp"
XI_PRM_CHIP_TEMP or "chip_temp"
XI_PRM_HOUS_TEMP or "hous_temp"
XI_PRM_HOUS_BACK_SIDE_TEMP or "hous_back_side_temp"

Writing Applications with xiAPI

Default parameters

After camera is opened by xiOpenDevice the default camera parameters are set by API. The default parameters might be different in different API versions. In order to ensure that your application will have camera in expected state with any API version - please set all parameters expected by your application to required value.

xiAPI Functions.

xiOpenDevice

Description:

This function initializes the device and returns a device handle.

Parameters:

- *DevId* - index of the device
- *hDevice* - handle to device

C Prototype:

```
XI_RETURN xiOpenDevice(IN DWORD DevId, OUT PHANDLE * hDevice);
```

xiOpenDeviceBy

Description:
This function initializes the device and returns a device handle. Device is selected according to used enumerator.
Parameters:

- *sel* - select method to be used for camera selection
- *prm* - string to identify device
- *hDevice* - handle to device

C Prototype:

```
XI_RETURN __cdecl xiOpenDeviceBy(IN XI_OPEN_BY sel, IN const char* prm, OUT PHANDLE hDevice);
```

xiCloseDevice

Description:
This function will un-initialize the specified device, closes its handle and releases allocated resources.
Parameters:

- *hDevice* - handle to device

C Prototype:

```
XI_RETURN xiCloseDevice(IN HANDLE hDevice);
```

xiGetNumberDevices

Description: This function enumerates all devices connected and returns the number of discovered devices. It is needed to be called before any other function of API is called by application.
Parameters:

- *pNumberDevices* - number of discovered devices

C Protopype:

```
XI_RETURN xiGetNumberDevices(OUT DWORD *pNumberDevices);
```

xiStartAcquisition

Description:
This function starts the data acquisition on the devices specified by the handle.
Parameters:

- *hDevice* - handle to device

C Prototype:

```
XI_RETURN xiStartAcquisition(IN HANDLE hDevice);
```

xiStopAcquisition

Description:
Ends the work cycle of the camera, stops data acquisition and deallocates internal image buffers.
Parameters:

- *hDevice* - handle to device

C Prototype:

```
XI_RETURN xiStopAcquisition(IN HANDLE hDevice);
```

xiGetImage

Description:
This function waits for next image is available at transport buffer. If available - it does all required image processing (unpack, sensor-defect-correction, demosaic) and fills image information to structure at img parameter. Note: Image processing is not done if XI_FRM_TRANSPORT_DATA is selected. In this case the function just set pointer to transport-buffer without any processing.
Parameters:

- *hDevice* - handle to device
- *TimeOut* - time interval required to wait for the image (in milliseconds).
- *img* - Pointer to image info structure

Note: Allocation of buffers is influenced by buffering policy. See details of behavior at parameter XI_PRM_BUFFER_POLICY.
C Prototype:

```
XI_RETURN xiGetImage(IN HANDLE hDevice, IN DWORD TimeOut,
                     INOUT XI_IMG * img);
```

The image structure XI_IMG description:

bytes	field name	description
4	size	Size of current structure on application side. When xiGetImage is called and size>=SIZE_XI_IMG_V2 then GPI_level, tsSec and tsUSec are filled.
ptr	bp	pointer to first pixel of image data
4	bp_size	Filled buffer size. When buffer policy is set to XI_BP_SAFE, xiGetImage will fill this field with current size of image data received.
4	frm	format of image data (same as in parameter XI_PRM_IMAGE_DATA_FORMAT)
4	width	width of image in pixels
4	height	height of image in pixels
4	nframe	frame number (reset by exposure, gain, downsampling change, auto exposure (AEAG))
4	tsSec	TimeStamp in seconds *1
4	tsUSec	TimeStamp in microseconds *1
4	GPI_level	Level of Digital inputs/outputs sampled at the start/finish of exposure. See *3
4	black_level	Black level of image (ONLY for MONO and RAW formats) *2
4	padding_x	Number of extra bytes provided at the end of each line to facilitate image alignment in buffers
4	AbsoluteOffsetX	Horizontal offset of origin of sensor and buffer image first pixel
4	AbsoluteOffsetY	Vertical offset of origin of sensor and buffer image first pixel
4	exposure_time_us	Exposure time of this image in microseconds. Note: Some camera models (MQ,MU) might report this exposure time earlier before exposure is applied to image.
4	gain_db (float)	Gain used for this image in dB (only valid for MQ,MD,MR camera families)
4	acq_nframe	Frame number. Reset only by acquisition start. NOT reset by change of exposure, gain, downsampling, auto exposure (AEAG).
4	image_user_data	data controlled by user application using XI_PRM_IMAGE_USER_DATA parameter
20	exposure_sub_times_us	Sub exposure times in us for XI_TRG_SEL_MULTIPLE_EXPOSURES or HDR

1 *Note: On most cameras the TimeStamp is represented in 32 bit microsecond number. In image header the fields **tsSec** and **tsUSec resets automatically** to zero after 2^32 useconds (4294 seconds and 967296 micro seconds). The recent firmware and API updates (2014-12) increased the size of TimeStamp counter to 40 bit. Period with this update is approx. 305 hours (=2^40 us).
TimeStamp is **NOT** implemented on some cameras. xiMU (MU9). Image header contains only constant number instead of valid TimeStamp.
2 *Note: xiQ cameras reports calculated black_level. We do not guaranty the precision of black_level calculation above 50ms of exposure and/or gain above 3dB.
3 *Note:: Order of bits and sampling time is model dependent. See GPI_Level Sampling

GPI_level Sampling in Header

Following table shows sample time and bits order for each camera family. Bits marking:

- isN = input at exposure start,
- ieN = input at exposure
- x = reserved

N is same number as in values used in XI_PRM_GPI_SELECTOR.

Model	Sample Time	Bits of 4 bytes (MSB first)
CB, CB-X8G3, MX-X4G2	Start/End of Exp	[x x x x ie6 ie5 ie4 ie3][x x ie2 ie1 x x x x][x x x x is6 is5 is4 is3][x x is2 is1 x x x x]
MC, MX-X2G2	Start/End of Expo.	[x x x x x x x x][ie3 ie2 x ie1 x x x x][x x x x x x x x][is3 is2 x is1 x x x x]
MQ	End of Expo.	[x x x x x x x x][x x x x x x x x][x x x x x x x x][x x x x x x x ie1]

xiSetParam

Description:
This function configures device (see xiAPI Parameters below).
Parameters:

- *hDevice* - handle to device
- *prm* - parameter name string.
- *value* - value that should be set to parameter
- *size* - size of value
- *type* - data type of value

C Prototype:

```
XI_RETURN xiSetParam(IN HANDLE hDevice, IN CHAR * prm, IN VOID * value,
                     IN DWORD size, IN XI_PRM_TYPE type);
```

xiGetDeviceInfoString

Description:
This function returns selected parameter of camera without opening it. It allows to quickly get information from each camera in multiple camera setups.
Parameters:

- *Devid* - index of the camera (same as on xiOpenDevice)

- *prm* - parameter name string
- *value* - pointer to result string
- *value_size* - size of string

C Prototype:

```
XIAPI XI_RETURN __cdecl xiGetDeviceInfoString(IN DWORD DevId, const char* prm, char* value, DWORD value_size);
```

Note: This function is capable to return only limited set of parameters:

- XI_PRM_DEVICE_SN
- XI_PRM_DEVICE_NAME
- XI_PRM_DEVICE_INSTANCE_PATH
- XI_PRM_DEVICE_LOCATION_PATH
- XI_PRM_DEVICE_TYPE

xiGetParam

Description:
This function returns parameters information (current value, minimum, maximum)(see xiAPI Parameters below).
Parameters:

- *hDevice* - handle to device
- *prm* - parameter name string
- *value* - value buffer where result will be stored
- *size* - size of value buffer
- *type* - expected value type

C Prototype:

```
XI_RETURN xiGetParam(IN HANDLE hDevice, IN CHAR * prm, IN VOID * value,
                     INOUT DWORD * size, OUT XI_PRM_TYPE * type);
```

xiAPI Parameters.

Each parameter contains of:

- Description: Describing the parameter behavior
- Type: Data type used internally for modeling parameter
- Default: Describing typical default value, however it can be different for different cameras
- Invalidators: List of parameters. Changing of any of those might lead to change of value or range (min, max, increment) of parameter described.
- Usage: Example of usage of this parameter in application (C code)

Device info parameters.

XI_PRM_DEVICE_NAME or “device_name”

Description: Returns device name.
Type: String.
Default: -
Usage:

```
xiGetParamString(handle, XI_PRM_DEVICE_NAME, char *, size);
```

XI_PRM_DEVICE_INSTANCE_PATH or "device_inst_path"

Description: Returns device instance path in operating system.
Type: String.
Default: -
Usage:

```
xiGetParamString(handle, XI_PRM_DEVICE_INSTANCE_PATH, path, sizeof(path));
```

XI_PRM_DEVICE_LOCATION_PATH or "device_loc_path"

Description: Returns device location path in operating system. It should reflect the connection position.
Type: String.
Default: -
Usage:

```
xiGetParamString(handle, XI_PRM_DEVICE_LOCATION_PATH, path, sizeof(path));
```

XI_PRM_DEVICE_TYPE or “device_type”

Description: Returns device type (1394, USB2.0, CURRERA…..).

Type: String.
Default: -
Usage:

```
xiGetParamString(handle, XI_PRM_DEVICE_TYPE, char*, size);
```

XI_PRM_DEVICE_MODEL_ID or “device_model_id”

Description: Returns the device model id.
Type: Integer.
Default: -
Usage:

```
xiGetParamInt(handle, XI_PRM_DEVICE_MODEL_ID, &model_id);
```

XI_PRM_SENSOR_MODEL_ID or “sensor_model_id”

Description: Returns the device sensor model id.
Type: Integer.
Default: -
Usage:

```
xiGetParamInt(handle, XI_PRM_SENSOR_MODEL_ID, &sensor_id);
```

XI_PRM_DEVICE_SN or “device_sn”

Description: Returns device serial number. Only string form is possible. It might contain also alphabet characters.
Type: String
Default: -
Usage:

```
char sn[20]="";
xiGetParamString(handle, XI_PRM_DEVICE_SN, sn, sizeof(sn));
```

Example:

```
// show serial number of this camera
char sn[20]="";
xiGetParamString(handle, XI_PRM_DEVICE_SN, sn, sizeof(sn));
printf("Serial number of the camera is: %s\n",sn);
```

XI_PRM_DEVICE_SENS_SN or device_sens_sn

Description: Returns sensor serial number.
Type: String.
Default: -
Usage:

```
char sens_sn[100]="";
xiGetParamString(handle, XI_PRM_DEVICE_SENS_SN, sens_sn, 100);
```

XI_PRM_DEVICE_USER_ID or "device_user_id"

Description: Get/Set custom user ID stored in camera non volatile memory. This can be later used as handle for opening or identification.
Type: String (ASCII).
Note: Available currently on certain models

Model	Maximum length
xiQ	10
xiB, xiT	4

Default: -
Usage:

```
xiGetParamString(handle,XI_PRM_DEVICE_USER_ID,name,sizeof(name));
```

Example of setting and retrieving the DEVICE_USER_ID

XI_PRM_DEVICE_MANIFEST or "device_manifest"

Description: Get XML of current device parameters and capabilities.

Type: String.
Note: Available currently on xiQ USB3.0, xiB, xiT cameras.
Default: -
Usage:

```
xiGetParamString(handle,XI_PRM_DEVICE_MANIFEST,value,sizeof(name));
```

XI_PRM_DEBUG_LEVEL or 'debug_level'

Description: Setting the API debug level allows to select amount of messages stored to debug output.

- In Windows use DebugView to view the current messages.
- In Linux the messages are printed to stderr

Type: Integer.

type	representing value	result
XI_DL_DETAIL	0	Prints same as XI_DL_TRACE plus locking of resources
XI_DL_TRACE	1	Prints errors, warnings and important informations
XI_DL_WARNING	2	Prints all errors and warnings
XI_DL_ERROR	3	Prints all errors
XI_DL_FATAL	4	Prints only important errors
XI_DL_DISABLED	100	Prints no messages

Default: XI_DL_WARNING
Usage:

```
xiSetParamInt(0, XI_PRM_DEBUG_LEVEL, XI_DL_TRACE);
```

XI_PRM_AUTO_BANDWIDTH_CALCULATION or 'auto_bandwidth_calculation'

Description: Setting this parameter the application can control API behavior. Setting to XI_OFF - API will skip auto bandwidth measuring and calculation before opening the camera (xiOpenDevice). Setting to XI_ON the measurement is enabled (default).
Note: It is important to set this parameter to XI_OFF in case when multiple cameras are connected to one hub with enabled acquisition and new camera should be opened - to not affect overall streaming by auto bandwidth measurement.
Type: Integer.
Default: XI_ON
Usage:

```
xiSetParamInt(0, XI_PRM_AUTO_BANDWIDTH_CALCULATION, XI_OFF);
```

XI_PRM_NEW_PROCESS_CHAIN_ENABLE or 'new_process_chain_enable'

Description: Setting this parameter the application can control API behavior. When set to XI_OFF - API will use original processing in image pipe for cameras families MU, MQ, MD. Setting to XI_ON - API will use newer processing type.
Note: There are some differences between processing so the switching may be done with caution. For older implementation we advise to stick to original processing. Only if some features require the newer processing it might be enabled. Switching may be done before xiOpenDevice.
Type: Integer.
Default: XI_OFF
Usage:

```
xiSetParamInt(0, XI_PRM_NEW_PROCESS_CHAIN_ENABLE, XI_ON);
```

Device acquisition parameters.

XI_PRM_EXPOSURE or “exposure”

Description: Sets exposure time in microseconds.
Type: Integer.
Default: 0
Usage:

```
xiSetParamInt(handle, XI_PRM_EXPOSURE, time_in_us);
```

XI_PRM_EXPOSURE_BURST_COUNT or "exposure_burst_count"

Description: Sets the number of times of exposure in one frame.
To finish exposure burst change this parameter to 1 and exposure will be finished by next trigger.
Note: This setting is valid only if the trigger selector is set to ExposureActive or ExposureStart.
Type: Integer.
Default: 1
Usage:

```
xiSetParamInt(handle, XI_PRM_EXPOSURE_BURST_COUNT, 5);
```

XI_PRM_GAIN_SELECTOR or “gain_selector”

Description: Selects type of gain for XI_PRM_GAIN. On some cameras there is possibility to select analog or digital gain separately.
Type: Integer.
Default: All
Usage:

```
xiSetParamInt(handle, XI_PRM_GAIN_SELECTOR, XI_GAIN_SELECTOR_ANALOG_ALL );
```

XI_PRM_GAIN or “gain”

Description: Sets gain in dB.
Type: Floating.
Default: 0
Usage:

```
xiSetParamFloat(handle, XI_PRM_GAIN, gain_in_db );
```

XI_PRM_DOWNSAMPLING or “downsampling”

Description: Changes image resolution by binning or skipping.
Parameter downsampling_rate controls the mapping of sensor pixels to output data.
Type: Integer.

downsampling_rate	result image
1	1 sensor pixel = 1 image pixel
2	2x2 sensor pixels = 1 image pixel
4	4x4 sensor pixels = 1 image pixel

Default: 1
Note: Downsampling can be changed only before an acquisition is started.
Usage:

```
xiSetParamInt(handle, XI_PRM_DOWNSAMPLING, 2);
```

Note: Changing this parameter will flush all images from the buffer queue.

XI_PRM_DOWNSAMPLING_TYPE or “downsampling_type”

Description: Changes image downsampling type (binning or skipping).
Type: Integer.

type	value	result image
XI_BINNING	0	pixels are interpolated - better image
XI_SKIPPING	1	pixels are skipped - higher frame rate

Default: 0 (XI_BINNING)
Usage:

```
xiSetParamInt(handle, XI_PRM_DOWNSAMPLING_TYPE, XI_SKIPPING);
```

Note: Changing this parameter will remove all images from the buffer queue.

XI_PRM_BINNING_SELECTOR or “binning_selector”

Description: Selects Binning engine to configure.
Type: Integer.

type	value	binning engine
XI_BIN_SELECT_SENSOR	0	on-chip image sensor binning engine is used (if supported)

Default: 0 (XI_BIN_SELECT_SENSOR)
Usage:

```
xiSetParamInt(handle, XI_PRM_BINNING_SELECTOR, XI_BIN_SELECT_SENSOR);
```

XI_PRM_BINNING_VERTICAL_MODE or “binning_vertical_mode”

Description: Sets the mode to use to combine vertical pixel together.
Type: Integer.

XI_BIN_MODE	value	binning engine
XI_BIN_MODE_SUM	0	The response from the combined pixels will be added, resulting in increased sensitivity.

XI_BIN_MODE_AVERAGE	1	The response from the combined pixels will be averaged, resulting in increased signal/noise ratio.

Default: 0 (XI_BIN_MODE_SUM)
Usage:

```
xiSetParamInt(handle, XI_PRM_BINNING_VERTICAL_MODE, XI_BIN_MODE_AVERAGE);
```

XI_PRM_BINNING_VERTICAL or "binning_vertical"

Description: Defines number of vertical photo-sensitive cells to combine.
Type: Integer.
Default: 1
Usage:

```
xiSetParamInt(handle, XI_PRM_BINNING_VERTICAL, 2);
```

Note: Setting this feature may automatically change other Binning/Decimation parameters in order to achieve a valid combination.
Note: This feature influences image size + sets ROI to default full image values.

XI_PRM_BINNING_HORIZONTAL_MODE or "binning_horizontal_mode"

Description: Sets the mode to use to combine horizontal pixel together.
Type: Integer.

XI_BIN_MODE	value	binning engine
XI_BIN_MODE_SUM	0	The response from the combined pixels will be added, resulting in increased sensitivity.
XI_BIN_MODE_AVERAGE	1	The response from the combined pixels will be averaged, resulting in increased signal/noise ratio.

Default: 0 (XI_BIN_MODE_SUM)
Usage:

```
xiSetParamInt(handle, XI_PRM_BINNING_HORIZONTAL_MODE, XI_BIN_MODE_AVERAGE);
```

XI_PRM_BINNING_HORIZONTAL or "binning_horizontal"

Description: Defines number of horizontal photo-sensitive cells to combine.
Type: Integer.
Default: 1
Usage:

```
xiSetParamInt(handle, XI_PRM_BINNING_HORIZONTAL, XI_BIN_MODE);
```

Note: Setting this feature may automatically change other Binning/Decimation parameters in order to achieve a valid combination.
Note: This feature influences image size + sets ROI to default full image values.

XI_PRM_BINNING_HORIZONTAL_PATTERN or "binning_horizontal_pattern"

Description: Defines binning horizontal pattern.
Type: Integer.

type	value	result image
XI_BIN_MONO	1	adjacent pixels are combined
XI_BIN_BAYER	2	Bayer pattern is preserved during pixel combining

Default: XI_BIN_MONO (for Monochorme cameras) / XI_BIN_BAYER (for Color cameras)
Usage:

```
xiSetParamInt(handle, XI_PRM_BINNING_HORIZONTAL_PATTERN, XI_BIN_BAYER);
```

Note: Setting this feature may automatically change other Binning/Decimation parameters in order to achieve a valid combination.
Note: This feature influences image size + sets ROI to default full image values.

XI_PRM_BINNING_VERTICAL_PATTERN or "binning_vertical_pattern"

Description: Defines binning vertical pattern.
Type: Integer.

type	value	result image
XI_BIN_MONO	1	adjacent pixels are combined
XI_BIN_BAYER	2	Bayer pattern is preserved during pixel combining

Default: XI_BIN_MONO (for Monochorme cameras) / XI_BIN_BAYER (for Color cameras)
Usage:

```
xiSetParamInt(handle, XI_PRM_BINNING_VERTICAL_PATTERN, XI_BIN_BAYER);
```

Note: Setting this feature may automatically change other Binning/Decimation parameters in order to achieve a valid combination.
Note: This feature influences image size + sets ROI to default full image values.

XI_PRM_DECIMATION_SELECTOR or “decimation_selector”

Description: Selects Decimation engine to configure.
Type: Integer.

type	value	binning engine
XI_DEC_SELECT_SENSOR	0	on-chip image sensor decimation engine is used (if supported)

Default: 0 (XI_DEC_SELECT_SENSOR)
Usage:

```
xiSetParamInt(handle, XI_PRM_DECIMATION_SELECTOR, XI_DEC_SELECT_SENSOR);
```

XI_PRM_DECIMATION_VERTICAL or “decimation_vertical”

Description: Reduces the vertical resolution of the image by the specified decimation factor.
Type: Integer.
Default: 1
Usage:

```
xiSetParamInt(handle, XI_PRM_DECIMATION_VERTICAL, decimation_vertical);
```

Note: Setting this feature may automatically change other Binning/Decimation parameters in order to achieve a valid combination.
Note: This feature influences image size + sets ROI to default full image values.

XI_PRM_DECIMATION_HORIZONTAL or “decimation_horizontal”

Description: Reduces the horizontal resolution of the image by the specified decimation factor.
Type: Integer.
Default: 1
Usage:

```
xiSetParamInt(handle, XI_PRM_DECIMATION_HORIZONTAL, decimation_horizontal);
```

Note: Setting this feature may automatically change other Binning/Decimation parameters in order to achieve a valid combination.
Note: This feature influences image size + sets ROI to default full image values.

XI_PRM_DECIMATION_HORIZONTAL_PATTERN or “decimation_horizontal_pattern”

Description: Defines decimation horizontal pattern.
Type: Integer.

type	value	result image
XI_DEC_MONO	1	adjacent pixels are decimated
XI_DEC_BAYER	2	Bayer pattern is preserved during pixel decimation

Default: XI_DEC_MONO (for Monochorme cameras) / XI_DEC_BAYER (for Color cameras)
Usage:

```
xiSetParamInt(handle, XI_PRM_DECIMATION_HORIZONTAL_PATTERN, XI_DEC_BAYER);
```

Note: Setting this feature may automatically change other Binning/Decimation parameters in order to set a valid combination.
Note: This feature influences image size + sets ROI to default full image values.

XI_PRM_DECIMATION_VERTICAL_PATTERN or “decimation_vertical_pattern”

Description: Defines decimation vertical pattern.
Type: Integer.

type	value	result image
XI_DEC_MONO	1	adjacent pixels are decimated
XI_DEC_BAYER	2	Bayer pattern is preserved during pixel decimation

Default: XI_DEC_MONO (for Monochorme cameras) / XI_DEC_BAYER (for Color cameras)
Usage:

```
xiSetParamInt(handle, XI_PRM_DECIMATION_VERTICAL_PATTERN, XI_DEC_BAYER);
```

Note: Setting this feature may automatically change other Binning/Decimation parameters in order to set a valid combination.
Note: This feature influences image size + sets ROI to default full image values.

XI_PRM_TEST_PATTERN_GENERATOR_SELECTOR or “test_pattern_generator_selector”

Description: Selects Test Pattern Generator Engine.
Type: Integer.

type	value	result image
XI_TESTPAT_GEN_SENSOR	0	Sensor Testpattern Generator
XI_TESTPAT_GEN_FPGA	1	FPGA Testpattern Generator

Default: 0 (XI_TESTPAT_GEN_SENSOR)
Usage:

```
xiSetParamInt(handle, XI_PRM_TEST_PATTERN_GENERATOR_SELECTOR , XI_TESTPAT_GEN_SENSOR);
```

XI_PRM_TEST_PATTERN or “test_pattern”

Description: Selects Test Pattern Type to be generated by the selected Generator Engine.
Type: Integer.

type	value	result image
XI_TESTPAT_OFF	0	Testpattern turned off.
XI_TESTPAT_BLACK	1	Image is filled with darkest possible image.
XI_TESTPAT_WHITE	2	Image is filled with brightest possible image.
XI_TESTPAT_GREY_HORIZ_RAMP	3	Image is filled horizontally with an image that goes from the darkest possible value to the brightest.
XI_TESTPAT_GREY_VERT_RAMP	4	Image is filled vertically with an image that goes from the darkest possible value to the brightest.
XI_TESTPAT_GREY_HORIZ_RAMP_MOVING	5	Image is filled horizontally with an image that goes from the darkest possible value to the brightest and moves from left to right.
XI_TESTPAT_GREY_VERT_RAMP_MOVING	6	Image is filled vertically with an image that goes from the darkest possible value to the brightest and moves from left to right.
XI_TESTPAT_HORIZ_LINE_MOVING	7	A moving horizontal line is superimposed on the live image.
XI_TESTPAT_VERT_LINE_MOVING	8	A moving vertical line is superimposed on the live image.
XI_TESTPAT_COLOR_BAR	9	Image is filled with stripes of color including White, Black, Red, Green, Blue, Cyan, Magenta and Yellow.
XI_TESTPAT_FRAME_COUNTER	10	A frame counter is superimposed on the live image.
XI_TESTPAT_DEVICE_SPEC_COUNTER	11	128bit counter.

Default: 0 (XI_TESTPAT_OFF)
Usage:

```
xiSetParamInt(handle, XI_PRM_TEST_PATTERN , XI_TESTPAT_OFF);
```

XI_PRM_SHUTTER_TYPE or "shutter_type"

Description: Type of sensor shutter.
Type: Integer.

type	value	result image
XI_SHUTTER_GLOBAL	0	Sensor Global Shutter(CMOS sensor)
XI_SHUTTER_ROLLING	1	Sensor Electronic Rolling Shutter(CMOS sensor)
XI_SHUTTER_GLOBAL_RESET_RELEASE	2	Sensor Global Reset Release Shutter(CMOS sensor)

Default: Depends on camera model
Invalidators: XI_PRM_TRG_SOURCE
Usage:

```
xiSetParamInt(handle, XI_PRM_SHUTTER_TYPE , shutter_type);
```

Note: This feature is available only on xiMU camera models.

XI_PRM_IMAGE_DATA_FORMAT or “imgdataformat”

Description: Format of image data returned by function xiGetImage. In order to simplify the control of the camera from application - the xiAPI automatically changes selected camera parameters and Image Processing after setting of XI_PRM_IMAGE_DATA_FORMAT. See Format Dependency Table.
Type: Integer.
Default: MONO8
Possible values:

Image Data Formats

value	**one pixel data in memory** [one_byte]
XI_MONO8	[Intensity] (Note1,Note2)
XI_MONO16	[Intensity LSB] [Intensity MSB] (Note1,Note2)
XI_RGB24	[Blue][Green][Red] (Note1)
XI_RGB32	[Blue][Green][Red][0]* (Note1)

XI_RGB_PLANAR	[Red][Red]...[Green][Green]...[Blue][Blue]... (Note1)
XI_RAW8	[pixel byte] raw data from transport (camera output)
XI_RAW16	[pixel byte low] [pixel byte high] 16 bits raw data from transport
XI_FRM_TRANSPORT_DATA	depends on data on the transport layer (Note3)

Note1: Higher CPU processing is required when this mode is selected because color filter array processing is implemented on PC. This processing is serialized when multiple cameras is used at once. The most effective way to get data from camera is to use XI_RAW8, where no additional processing is done in API.
Note2: On monochromatic cameras the black level is not subtracted in XI_MONO8 and XI_MONO16 formats by Image Processing in xiAPI, so black level remains the same as in RAW format.
Note3: When using Transport Data Format, the Image Processing block from Image Data Flow is skipped and therefore the Transport format is the most effective data format in terms of CPU and RAM usage.

XI_PRM_IMAGE_DATA_FORMAT_RGB32_ALPHA or "imgdataformatrgb32alpha"

Description: The alpha channel of RGB32 output image format(see XI_PRM_IMAGE_DATA_FORMAT).
Type: Integer.
Default: 0
Possible values:

Data Format Dependency Table

Following parameters and Image Processing are controlled **automatically** by setting of XI_PRM_IMAGE_DATA_FORMAT:

Format	Parameters controlled automatically	Image Processing
XI_MONO8	XI_PRM_SENSOR_DATA_BIT_DEPTH = 8* XI_PRM_OUTPUT_DATA_BIT_DEPTH = 8* XI_PRM_OUTPUT_DATA_PACKING = OFF	enabled
XI_RAW8	XI_PRM_SENSOR_DATA_BIT_DEPTH = 8* XI_PRM_OUTPUT_DATA_BIT_DEPTH = 8* XI_PRM_OUTPUT_DATA_PACKING = OFF	disabled
XI_MONO16**	XI_PRM_SENSOR_DATA_BIT_DEPTH = maximum XI_PRM_OUTPUT_DATA_BIT_DEPTH = SENSOR_DATA_BIT_DEPTH XI_PRM_OUTPUT_DATA_PACKING = ON*	enabled
XI_RAW16**	XI_PRM_SENSOR_DATA_BIT_DEPTH = maximum XI_PRM_OUTPUT_DATA_BIT_DEPTH = SENSOR_DATA_BIT_DEPTH XI_PRM_OUTPUT_DATA_PACKING = ON*	disabled
XI_RGB32 XI_RGB24 XI_RGB_PLANAR	XI_PRM_SENSOR_DATA_BIT_DEPTH = maximum XI_PRM_OUTPUT_DATA_BIT_DEPTH = SENSOR_DATA_BIT_DEPTH XI_PRM_OUTPUT_DATA_PACKING = ON*	enabled

Note (*): Only if camera implementation allows this mode.
Note (**): For XI_RAW16, XI_MONO16 the parameter XI_PRM_IMAGE_DATA_BIT_DEPTH will be equal to XI_PRM_OUTPUT_DATA_BIT_DEPTH. For other formats the XI_PRM_IMAGE_DATA_BIT_DEPTH will be 8.
After changing of XI_PRM_IMAGE_DATA_FORMAT the image resolution (XI_PRM_WIDTH,XI_PRM_HEIGHT) might change. Please check or set image resolution after changing of Image Data Format.
Bits alignment: Values are aligned to LSB.

sensor bits per pixel	values in modes XI_RAW16
10	0-1023
12	0-4095
14	0-16383

Example: Camera produces 10 bits data and data format XI_RAW16 bit is selected - each 16bit word (pixel) can contain values in range 0-1023.
Usage:

```
xiSetParamInt(handle, XI_PRM_IMAGE_DATA_FORMAT, XI_MONO8);
```

Note: For color modes **XI_RGB32** and **XI_RGB24** the image from sensor should be pre-processed. CPU load is higher in these modes. Setting this parameter will reset current region of interest. **XI_RGB24** is being processed from the **XI_RGB32** by removing the unused Alpha channel creating a slightly higher CPU load then the **XI_RGB32** format.

XI_PRM_IMAGE_PAYLOAD_SIZE or "imgpayloadsize"

Description: Buffer size in bytes sufficient for output image returned by xiGetImage
Type: Integer - Read only

XI_PRM_TRANSPORT_PIXEL_FORMAT or "transport_pixel_format"

Description: Transport pixel format is format of data transported by link to transport layer. It might be modified after setting of XI_PRM_IMAGE_DATA_FORMAT, XI_PRM_OUTPUT_DATA_PACKING, XI_PRM_OUTPUT_DATA_BIT_DEPTH, ...
Type: Enumerator - Enumerated by XI_GenTL_Image_Format_e
Default: (depends on camera model)

XI_PRM_FRAMERATE or “framerate”

Description: Defines frames per second of sensor. On some sensors it is possible to set framerate - see Note.
See more details in article Frame Rate Control.
Type: Float.
Default: Maximum possible at current exposure.
Invalidators: XI_PRM_IMAGE_DATA_FORMAT, XI_PRM_EXPOSURE, XI_PRM_ACQ_TIMING_MODE, XI_PRM_LIMIT_BANDWIDTH, XI_PRM_SENSOR_TAPS, XI_PRM_DOWNSAMPLING, XI_PRM_DOWNSAMPLING_TYPE, XI_PRM_HEIGHT
Usage:

```
float fps=0;
xiGetParamFloat(handle, XI_PRM_FRAMERATE, &fps);
```

Note: On some camera models it is possible to change or limit acquisition frame rate. Frame rate value should be within possible range. Use following code to get range.

```
xiGetParamFloat(handle, XI_PRM_FRAMERATE XI_PRM_INFO_MIN, &min_fps);
xiGetParamFloat(handle, XI_PRM_FRAMERATE XI_PRM_INFO_MAX, &max_fps);
```

Use following code to set 10 FPS on MQ, MD cameras:

```
xiSetParamInt(h,XI_PRM_ACQ_TIMING_MODE, XI_ACQ_TIMING_MODE_FRAME_RATE);
xiSetParamFloat(h, XI_PRM_FRAMERATE, 10);
```

Use following code to limit frame rate 10 FPS on CB,MT,MX,MC cameras:

```
xiSetParamInt(h,XI_PRM_ACQ_TIMING_MODE, XI_ACQ_TIMING_MODE_FRAME_RATE_LIMIT);
xiSetParamFloat(h, XI_PRM_FRAMERATE, 10);
```

XI_PRM_BUFFER_POLICY or “buffer_policy”

Description: Defines buffer handling. Can be safe, data will be copied to user/app buffer or unsafe, user will get internally allocated buffer without data copy
Type: Integer.
Default: BP_UNSAFE.
Usage:

```
xiSetParamInt(handle, XI_PRM_BUFFER_POLICY, XI_BP_SAFE);
```

Note: Click to below link to open simple description of buffer policy.
buffer_policy_in_xiApi.png

XI_PRM_COUNTER_SELECTOR or “counter_selector”

Description: Selects which frame counter must be returned
Type: Integer.

counter selector	value	**meaning**
XI_CNT_SEL_TRANSPORT_SKIPPED_FRAMES	0	Number of skipped frames on transport layer (e.g. when an image gets lost while transmission). Occur when the capacity of transport channel does not allow to transfer all data.
XI_CNT_SEL_API_SKIPPED_FRAMES	1	Number of skipped frames on API layer. Occur when application does not process the images as quickly as they are received from the camera
XI_CNT_SEL_TRANSPORT_TRANSFERRED_FRAMES	2	Number of successfully transferred frames on the transport layer.
XI_CNT_SEL_FRAME_MISSED_TRIGGER_DUETO_OVERLAP*	3	Number of missed triggers due to overlap.
XI_CNT_SEL_FRAME_MISSED_TRIGGER_DUETO_FRAME_BUFFER_OVR*	4	Number of missed triggers due to frame buffer full.
XI_CNT_SEL_FRAME_BUFFER_OVERFLOW*	5	Frame buffer full counter.

Note (*): available only on PCIe cameras (xiX, xiB, xiT cameras.)
Default: XI_CNT_SEL_TRANSPORT_SKIPPED_FRAMES.
Usage:

```
xiSetParamInt(handle, XI_PRM_COUNTER_SELECTOR, XI_CNT_SEL_TRANSPORT_SKIPPED_FRAMES);
```

Note: Could be returned number of skipped frames on the transport layer, number of skipped frames on API layer, number of successfully transferred frames.

XI_PRM_COUNTER_VALUE or “counter_value”

Description: Returns value of selected (by XI_PRM_COUNTER_SELECTOR) frame counter.
Type: Integer.
Default: 0.
Usage:

```
int number_of_skipped_frames = 0;
xiSetParamInt(handle, XI_PRM_COUNTER_SELECTOR, XI_CNT_SEL_API_SKIPPED_FRAMES);
```

```
xiGetParamInt(handle, XI_PRM_COUNTER_VALUE, &number_of_skipped_frames);
```

Note: All counters reset after acquisition stop.

XI_PRM_OFFSET_X or “offsetX”

Description: Horizontal offset from the origin to the area of interest (in pixels). The sum of XI_PRM_OFFSET_X and XI_PRM_WIDTH must be equal or lower than the image resolution(width). Number must be divisible by the minimum increment which can be read out using the api parameter modifier XI_PRM_INFO_INCREMENT
Type: Integer.
Default: 0
Invalidators: XI_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING
Usage:

```
xiSetParamInt(handle, XI_PRM_OFFSET_X, offset_x);
```

Note: Changing of this parameter will remove all images from buffer queue.

XI_PRM_OFFSET_Y or “offsetY”

Description: Vertical offset from the origin to the area of interest (in pixels). The sum of XI_PRM_OFFSET_Y and XI_PRM_HEIGHT must be equal or lower than the image resolution(height). Number must be divisible by the minimum increment which can be read out using the api parameter modifier XI_PRM_INFO_INCREMENT
Type: Integer.
Default: 0
Invalidators: XI_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING
Usage:

```
xiSetParamInt(handle, XI_PRM_OFFSET_Y, offset_y);
```

Note: Changing of this parameter will remove all images from buffer queue.

XI_PRM_WIDTH or “width”

Description:

- If camera runs in single region mode: this parameter is width of the image provided by the device (in pixels). The sum of XI_PRM_OFFSET_X and XI_PRM_WIDTH must be equal or lower than the image resolution(width). Number must be divisible by the minimum increment which can be read out using the api parameter modifier XI_PRM_INFO_INCREMENT
- If camera runs in multiple region mode (XI_PRM_REGION_SELECTOR): this parameter is width of region currently selected (in pixels).
  Type: Integer.
  Default: Full resolution width.
  Invalidators: XI_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING, XI_PRM_IMAGE_DATA_FORMAT, XI_PRM_OUTPUT_DATA_PACKING, XI_PRM_OUTPUT_DATA_BIT_DEPTH, XI_PRM_HEIGHT

Usage:

```
xiSetParamInt(handle, XI_PRM_WIDTH, image_width);
```

Note: Changing of this parameter will remove all images from buffer queue.

XI_PRM_HEIGHT or “height”

Description:

- If camera runs in single region mode: this parameter is height of the image provided by the device (in pixels). The sum of XI_PRM_OFFSET_Y and XI_PRM_HEIGHT must be equal or lower than the image resolution(height). Number must be divisible by the minimum increment which can be read out using the api parameter modifier XI_PRM_INFO_INCREMENT
- If camera runs in multiple region mode (XI_PRM_REGION_SELECTOR): this parameter is height of region currently selected.
  Type: Integer.
  Default: Full resolution height.
  Invalidators: XI_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING, XI_PRM_REGION_SELECTOR, XI_PRM_IMAGE_DATA_FORMAT, XI_PRM_OUTPUT_DATA_PACKING, XI_PRM_OUTPUT_DATA_BIT_DEPTH, XI_PRM_WIDTH

Usage:

```
xiSetParamInt(handle, XI_PRM_HEIGHT, image_height);
```

Note 1: Changing of this parameter will remove all images from buffer queue.
Note 2: In case of using small ROI in combination with Fresco FL1100 controller, please read this article.

XI_PRM_REGION_SELECTOR or "region_selector"

Description: Selects Region in Multiple ROI. Parameters: XI_PRM_WIDTH, XI_PRM_HEIGHT, XI_PRM_OFFSET_X, XI_PRM_OFFSET_Y, XI_PRM_REGION_MODE are related to the particular region
Type: Integer.
Default: 0.
Invalidators: XI_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING
Usage:

```
xiSetParamInt(handle, XI_PRM_REGION_SELECTOR , selector);
```

Note: width and offset_x could be changed only for Region 0
Note2: Regions has to be in order from top to bottom. Region N has to start after Region N-1 ends.

XI_PRM_REGION_MODE or "region_mode"

Description: Activates/deactivates Region selected by Region Selector in Multiple ROI.
Type: Integer.
Default: 1 for Region selector 0 and 0 for all other regions
Invalidators: XI_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING
Usage:

```
xiSetParamInt(handle, XI_PRM_REGION_MODE , mode);
```

XI_PRM_HORIZONTAL_FLIP or "horizontal_flip"

Description: Activates horizontal flip if available in camera. Currently available only some MQ cameras models.
Note: Requires XI_PRM_NEW_PROCESS_CHAIN_ENABLE set to XI_ON.
Type: Integer.
Default: 0 for disabled flipping
Usage:

```
xiSetParamInt(handle, XI_PRM_HORIZONTAL_FLIP , 1);  //enable horizontal flipping
```

XI_PRM_VERTICAL_FLIP or "vertical_flip"

Description: Activates vertical flip if available in camera. Currently available only some MQ cameras models.
Note: Requires XI_PRM_NEW_PROCESS_CHAIN_ENABLE set to XI_ON.
Type: Integer.
Default: 0 for disabled flipping
Usage:

```
xiSetParamInt(handle, XI_PRM_VERTICAL_FLIP, 1);  //enable horizontal flipping
```

XI_PRM_LUT_EN or “LUTEnable”

Description: Activates Look-Up-Table (LUT).
Type: Integer.
Default: 0ff(0).

lut_state	operation
0	sensor pixels are transferred directly
1	sensor pixels are mapped through LUT

Usage:

```
xiSetParamInt(handle, XI_PRM_LUT_EN, lut_state);
```

Note: LUT parameters are valid only for some cameras. E.g. CURRERA-R and xiQ supports LUT. xiMU (MU9PM-MH) does NOT support it.
Note2: For xiQ cameras setting XI_PRM_LUT_EN also uploads previously set values in to camera. Values are latched in API.

XI_PRM_LUT_INDEX or “LUTIndex”

Description: Controls the index (offset) of the coefficient to access in the LUT.
Type: Integer.
Default: 0
Usage:

```
xiSetParamInt(handle, XI_PRM_LUT_INDEX, lut_index);
```

Note: Range of applicable indexes depends on sensor digitization bit depth (sensor_bit_depth). It could be obtained:

```
xiGetParam(handle, XI_PRM_LUT_INDEX XI_PRM_INFO_MIN, &minimum_lut_index);
xiGetParam(handle, XI_PRM_LUT_INDEX XI_PRM_INFO_MAX, &maximum_lut_index);
```

All xiQ cameras have lut "N-bit to N-bit". Currera-R features LUT N-bit to 8-bit".

XI_PRM_LUT_VALUE or “LUTValue”

Description: Defines value at entry LUTIndex of the LUT.
Type: Integer.

Default: 0
Usage:

```
xiSetParamInt(handle, XI_PRM_LUT_ VALUE, lut_value_for_index);
```

Note: Range of applicable values depends on sensor digitization bit depth (sensor_bit_depth). It could be obtained:

```
xiGetParam(handle, XI_PRM_LUT_VALUE XI_PRM_INFO_MIN, &minimum_lut_index);
xiGetParam(handle, XI_PRM_LUT_VALUE XI_PRM_INFO_MAX, &maximum_lut_index);
```

All xiQ cameras have lut "N-bit to N-bit". Currera-R features LUT N-bit to 8-bit".
Note: For xiQ cameras setting values has no direct effect on image, only after setting XI_PRM_LUT_EN to 1, will apply all changes to camera.

XI_PRM_TRG_SOURCE or “trigger_source”

Description: Defines source of trigger.
Type: Integer.
Default: TRG_OFF
Possible values:

trigger_source	Camera operation
XI_TRG_OFF	Capture of next image is automatically started after previous.
XI_TRG_EDGE_RISING	Capture is started on rising edge of selected input.
XI_TRG_EDGE_FALLING	Capture is started on falling edge of selected input.
XI_TRG_SOFTWARE	Capture is started with software trigger. See here.
XI_TRG_LEVEL_HIGH	Specifies that the trigger is considered valid as long as the level of the source signal is high.
XI_TRG_LEVEL_LOW	Specifies that the trigger is considered valid as long as the level of the source signal is low.

Usage:

```
xiSetParamInt(handle, XI_PRM_TRG_SOURCE, trigger_source);
```

Note: To set input as external trigger, XI_PRM_GPI_MODE of selected input should be set to XI_GPI_TRIGGER. See example at XI_PRM_GPI_MODE.
Example:

```
// enable trigger by software
xiSetParamInt(handle, XI_PRM_TRG_SOURCE, XI_TRG_SOFTWARE);
// start acquisition
xiStartAcquisition(handle);
int frames=5;
while (frames--)
{
  // trigger next image
  int trigger_retry=1000;
  while(--trigger_retry)
  {
    if (XI_OK == xiSetParamInt(handle, XI_PRM_TRG_SOFTWARE, 1))
      break;
    // on some cameras (CURRERA-R) the TRG software
    // returns error in case if it is too early
    // to start next exposure
    Sleep(1);
  }
  // get the image
  xiGetImage(handle, 1000, &image);
}
```

For more examples see xiAPI Camera Trigger and Synchronization Signals.

XI_PRM_TRG_SOFTWARE or “trigger_software”

Description: Generates an internal trigger. XI_PRM_TRG_SOURCE have be set to TRG_SOFTWARE.
Type: Integer.
Default: -
Usage:

```
xiSetParamInt(handle, XI_PRM_TRG_SOFTWARE, 0);
```

Note: In several models xiMU (MU9), xiQ, Currera R this is active even if XI_PRM_TRG_SOURCE is set to external signal.

XI_PRM_TRG_DELAY or "trigger_delay"

Description: When set delay time is inserted between camera trigger input and activating sensor integration. Delay time is set in us.
Type: Integer.
Default: 0us
Invalidators: XI_PRM_TRG_SELECTOR
Usage:

```
xiSetParamInt(handle, XI_PRM_TRG_SELECTOR , XI_TRG_SEL_FRAME_START);
xiSetParamInt(handle, XI_PRM_TRG_DELAY , 5000);  //5ms delay
```

Note: Setting of this parameter is applicable for selected cameras:

- xiX, xiB, xiT, xiC
- xiMU (MU9). Granularity of real delay duration depends on sensor settings (line read out time). Typical granularity is up to 100 microseconds. Maximum time is approx. 100ms for xiMU camera.

XI_PRM_AVAILABLE_BANDWIDTH "available_bandwidth"

Description: Calculate and return available interface bandwidth. Unit is Megabits (1000000) per sec.
Type: Integer.
Default: Maximum of given interface
Usage:

```
xiGetParamInt(handle, XI_PRM_AVAILABLE_BANDWIDTH, &interface_data_rate);
```

XI_PRM_LIMIT_BANDWIDTH "limit_bandwidth"

Description: Camera acquisition data-rate Limit on transport layer in Megabits (1000000) per second. API controls the camera clock or increases the line period [1] in order to achieve the closest data-rate as the Limit value set, ensuring the data-rate is below the Limit. This parameter can be used to decrease data-rate e.g. when more cameras are connected to same interface to share same channel. In order to activate the limit - application should set also XI_PRM_LIMIT_BANDWIDTH_MODE = XI_ON, see example below.
Type: Integer.
Default: Maximum of given camera
Note [1]: Controlling method (clock or line period) depends on the camera model.
Usage:

```
xiSetParamInt(handle, XI_PRM_LIMIT_BANDWIDTH , datarate_in_Mbits_sec);
```

Example:

```
// get interface data rate
int interface_data_rate_mbps=2500;
// calculate datarate for each camera
#define CONNECTED_CAMERAS_TO_SAME_HUB 3
int camera_data_rate = interface_data_rate_mbps / CONNECTED_CAMERAS_TO_SAME_HUB;
#define MARGIN_MBitsPER_SECOND 300
camera_data_rate -= MARGIN_MBitsPER_SECOND;
// set data rate
xiSetParamInt(handle, XI_PRM_LIMIT_BANDWIDTH , camera_data_rate);
// enable the limiting
xiSetParamInt(handle, XI_PRM_LIMIT_BANDWIDTH_MODE , XI_ON);
```

See more at our application note about Multiple Cameras Setup

XI_PRM_LIMIT_BANDWIDTH_MODE "limit_bandwidth_mode"

Description: Controls if the XI_PRM_LIMIT_BANDWIDTH is active. When disabled, lower level specific features are expected to control the throughput. When enabled, XI_PRM_LIMIT_BANDWIDTH controls the overall throughput.*Type:* Integer.
Default: ON (after auto bandwidth measurement)
Usage:

```
xiSetParamInt(handle, XI_PRM_LIMIT_BANDWIDTH_MODE , XI_OFF);
```

XI_PRM_SENSOR_TAPS "sensor_taps"

Description: Set/get the number of taps used on sensor.
Type: Integer.
Default: 1
Usage:

```
xiSetParamInt(handle, XI_PRM_SENSOR_TAPS ,number_of_taps);
```

XI_PRM_SENSOR_CLOCK_FREQ_HZ "sensor_clock_freq_hz"

Description: Set or return the sensor clock frequency. This clock is specific to sensor used. See documentation/application for the camera to use this parameter.
Type: Float.
Default: (depends on sensor)
Invalidators: XI_PRM_LIMIT_BANDWIDTH
Usage:

```
xiSetParamInt(handle, XI_PRM_SENSOR_CLOCK_FREQ_HZ, clock_freq);
```

XI_PRM_SENSOR_CLOCK_FREQ_INDEX "sensor_clock_freq_index"

Description: Sensor clock frequency. Selects frequency on cameras which supports only some specific frequencies.
Type: Int.
Default: (depends on sensor)
Invalidators: XI_PRM_LIMIT_BANDWIDTH
Usage:

```
xiSetParamInt(handle, XI_PRM_SENSOR_CLOCK_FREQ_INDEX, clock_id);
```

XI_PRM_SENSOR_OUTPUT_CHANNEL_COUNT "sensor_output_channel_count"

Description: Number of output channels from sensor used for data transfer.
Type: Int.
Default: (depends on sensor)
Invalidators:
Usage:

```
int count=0;
xiGetParamInt(handle, XI_PRM_SENSOR_OUTPUT_CHANNEL_COUNT, &count);
```

XI_PRM_SENSOR_DATA_BIT_DEPTH "sensor_bit_depth"

Description: Returns the bit depth of the pixel data received from sensor.
Type: Int.
Default: (depends on sensor)
Invalidators: XI_PRM_IMAGE_DATA_FORMAT
Usage:

```
xiGetParamInt(handle,XI_PRM_SENSOR_DATA_BIT_DEPTH ,&sensor_bit_depth);
```

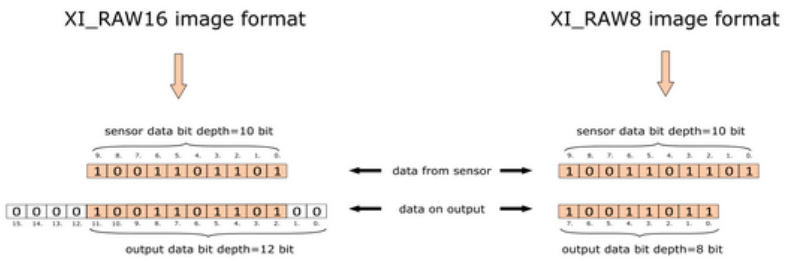
Read more at XiAPI Image Data Flow.

XI_PRM_OUTPUT_DATA_BIT_DEPTH "output_bit_depth"

Description: The bit depth of the output data from camera (=transport layer).
Type: Int.
Default: (depends on camera and sensor)
Invalidators: XI_PRM_IMAGE_DATA_FORMAT
Usage:

```
xiGetParamInt(handle,XI_PRM_OUTPUT_DATA_BIT_DEPTH ,&sensor_bit_depth);
```

Read more at XiAPI Image Data Flow.



XI_PRM_IMAGE_DATA_BIT_DEPTH "image_data_bit_depth"

Description: The bit depth of the pixel data returned by function xiGetImage. If MONO16 or RAW16 image formats are used this parameter defines alignment of the data on the xiGetImage.
Type: Int.
Default: (depends on sensor)
Invalidators: XI_PRM_IMAGE_DATA_FORMAT, XI_PRM_OUTPUT_DATA_BIT_DEPTH
Usage:

```
xiGetParamInt(handle,XI_PRM_IMAGE_DATA_BIT_DEPTH ,&image_data_bit_depth);
```

Read more at XiAPI Image Data Flow.

XI_PRM_OUTPUT_DATA_PACKING "output_bit_packing"

Description: This feature enables bit packing on transport data layer (USB3), thus increasing the maximum frame rate when data with 10 bits per pixel is transported. For more info please see Transport Data Packing feature description.
Type: Int.
Default: XI_OFF
Invalidators: XI_PRM_IMAGE_DATA_FORMAT, XI_PRM_OUTPUT_DATA_BIT_DEPTH
Usage:

```
xiSetParamInt(handle,XI_PRM_OUTPUT_DATA_PACKING,XI_ON);
```

Read more at XiAPI Image Data Flow.

XI_PRM_OUTPUT_DATA_PACKING_TYPE "output_bit_packing_type"

Description: This feature chooses output data packing type(ximea grouping 10g160, 12g192, 14g224), PFNC packing 10p, 12p...). For more info please see Transport Data Packing feature description.
Type: Int.
Default: XI_DATA_PACK_XI_GROUPING
Invalidators: XI_PRM_IMAGE_DATA_FORMAT, XI_PRM_OUTPUT_DATA_BIT_DEPTH
Usage:

```
xiSetParamInt(handle,XI_PRM_OUTPUT_DATA_PACKING_TYPE,XI_DATA_PACK_XI_GROUPING);
```

XI_PRM_ACQ_TIMING_MODE "acq_timing_mode"

Description: This parameter defines the acquisition timing mode.
Type: Int.
Default: XI_ACQ_TIMING_MODE_FREE_RUN
Possible Values:

Values	Description
XI_ACQ_TIMING_MODE_FREE_RUN	camera acquires images at a maximum possible framerate
XI_ACQ_TIMING_MODE_FRAME_RATE	please refer to our Frame Rate Control support page
XI_ACQ_TIMING_MODE_FRAME_RATE_LIMIT	please refer to our Frame Rate Control support page

Usage:

```
xiSetParamInt(handle,XI_PRM_ACQ_TIMING_MODE,XI_ACQ_TIMING_MODE_FRAME_RATE);
```

XI_PRM_TRG_SELECTOR "trigger_selector"

Description: This parameter selects the type of trigger .
Type: Int.
Default: XI_TRG_SEL_FRAME_START
Possible Values:

Values	Description
XI_TRG_SEL_FRAME_START	the trigger defines the start of the frame exposure
XI_TRG_SEL_EXPOSURE_ACTIVE	please refer to our Exposure Defined by Trigger Pulse Length support page
XI_TRG_SEL_FRAME_BURST_START	please refer to our Frame Burst Modes support page
XI_TRG_SEL_FRAME_BURST_ACTIVE	please refer to our Frame Burst Modes support page
XI_TRG_SEL_MULTIPLE_EXPOSURES	Selects a trigger which when first trigger starts exposure and consequent pulses are gating exposure(active HI)
XI_TRG_SEL_EXPOSURE_START	Selects a trigger controlling the start of the exposure of one Frame.
XI_TRG_SEL_MULTI_SLOPE_PHASE_CHANGE	Selects a trigger controlling the multi slope phase in one Frame (phase0 -> phase1) or (phase1 -> phase2)

Usage:

```
xiSetParamInt(handle,XI_PRM_TRG_SELECTOR,XI_TRG_SEL_EXPOSURE_ACTIVE);
```

XI_PRM_ACQ_FRAME_BURST_COUNT "acq_frame_burst_count"

Description: Sets the number of frames to be acquired after trigger pulse has been sent to the camera. This setting is valid only if the trigger selector is set to **FrameBurstStart**. For more info please refer to our Frame Burst Modes support page. If burst count is set to zero (0) then number of acquired frames will not be limited (=endless).
Type: Int.
Default: 1
Usage:

```
xiSetParamInt(handle,XI_PRM_ACQ_FRAME_BURST_COUNT,3);
```

XI_PRM_IMAGE_USER_DATA "image_user_data"

Description: Sets the user data (32bit number) into camera. The following frame captured by camera will have this number stored at image header. The number is accessible later after xiGetImage in XI_IMG structure as image_user_data.
Type: Int.
Default: 0
Cameras supported: xiB, xiT
Usage:

```
uint32_t data=7;
xiSetParamInt(handle,XI_PRM_IMAGE_USER_DATA,data);
xiGetImage(handle,&image,5000);
printf("Image captured has user_data:%d\n",image.image_user_data);
```

Color management settings

XI_PRM_WB_KR or “wb_kr”

Description: White balance red coefficient.
Type: Float.
Default: 1.0
Usage:

```
xiSetParamFloat(handle, XI_PRM_WB_KR, wb_coef);
```

XI_PRM_WB_KG or “wb_kg”

Description: White balance green coefficient.
Type: Float.
Default: 1.0
Usage:

```
xiSetParamFloat(handle, XI_PRM_WB_KG, wb_coef);
```

XI_PRM_WB_KB or “wb_kb”

Description: White balance blue coefficient.
Type: Float.
Default: 1.0
Usage:

```
xiSetParamFloat(handle, XI_PRM_WB_KB, wb_coef);
```

XI_PRM_MANUAL_WB or “manual_wb”

Description: Manual white balance. Takes white balance from square in image center of next image received by xiGetImage. Square have 1/8th of width and height of image. The function expects white sheet of paper exposed to 50% of values (RGB values should be around 128). As result of setting of manual_wb three parameters are changed: "wb_kb", "wb_kg" and "wb_kr". User application can store them and recall when needed to set the white balance back.
Type: Integer.
Default: -
Usage:

```
// now camera should see the white color in approximately 50% of level
xiSetParamInt(handle, XI_PRM_MANUAL_WB, 1);
xiGetImage(handle, &image); // now API automatically calculates the white balance
xiGetImage(handle, &image); // this and next images will have corrected white balance
```

XI_PRM_AUTO_WB or “auto_wb”

Description: Automatic white balance.
Type: Integer.
Default: 0 (disabled)
Usage:

```
xiSetParamInt(handle, XI_PRM_AUTO_WB, 1);
```

XI_PRM_GAMMAY or “gammaY”

Description: Luminosity gamma.
Range: 0.3 (highest correction); 1 (no correction)
Type: Float.
Default: 0.47
Usage:

```
xiSetParamFloat(handle, XI_PRM_GAMMAY, gammaY);
```

XI_PRM_GAMMAC or “gammaC”

Description: Chromaticity gamma.
Type: Float.
Default: 0.8
Usage:

```
xiSetParamFloat(handle, XI_PRM_GAMMAC, gammaC);
```

XI_PRM_SHARPNESS or “sharpness”

Description: Sharpness Strength. The range is -4 (less sharp) to +4 (more sharp).
Type: Float.
Default: 0.0 (neutral)
Usage:

```
xiSetParamFloat(handle, XI_PRM_SHARPNESS, sharpness_level);
```

XI_PRM_CC_MATRIX_00 or “ccMTX00”

XI_PRM_CC_MATRIX_01 or “ccMTX01”

XI_PRM_CC_MATRIX_02 or “ccMTX02”

XI_PRM_CC_MATRIX_03 or “ccMTX03”

XI_PRM_CC_MATRIX_10 or “ccMTX10”

XI_PRM_CC_MATRIX_11 or “ccMTX11”

XI_PRM_CC_MATRIX_12 or “ccMTX12”

XI_PRM_CC_MATRIX_13 or “ccMTX13”

XI_PRM_CC_MATRIX_20 or “ccMTX20”

XI_PRM_CC_MATRIX_21 or “ccMTX21”

XI_PRM_CC_MATRIX_22 or “ccMTX22”

XI_PRM_CC_MATRIX_23 or “ccMTX23”

XI_PRM_CC_MATRIX_30 or “ccMTX30”

XI_PRM_CC_MATRIX_31 or “ccMTX31”

XI_PRM_CC_MATRIX_32 or “ccMTX32”

XI_PRM_CC_MATRIX_33 or “ccMTX33”

Description: Color Correction Matrix elements.
Type: Float.
Default:

coefficients					default values			
M_00	M_01	M_02	M_03	=	1.0	0.0	0.0	0.0
M_10	M_11	M_12	M_13	=	0.0	1.0	0.0	0.0
M_20	M_21	M_22	M_23	=	0.0	0.0	1.0	0.0
M_30	M_31	M_32	M_33	=	0.0	0.0	1.0	0.0

Usage:

```
xiSetParamFloat(handle, XI_PRM_CC_MATRIX_00, float);
```

XI_PRM_DEFAULT_CC_MATRIX or “defccMTX”

Description: Set default Color Correction Matrix.
Type: -.
Default: -
Usage:

```
xiSetParamFloat(handle, XI_PRM_DEFAULT_CC_MATRIX, 0);
```

XI_PRM_IMAGE_IS_COLOR or “iscolor”

Description: Returns 1 for color cameras.
Type: Integer.
Default: By default -.
Usage:

```
xiGetParam(handle, XI_PRM_IMAGE_IS_COLOR or "iscolor", &iscolor, sizeof(int), xiTypeInteger);
int xiGetParamInt(handle, XI_PRM_IMAGE_IS_COLOR, &iscolor);
```

XI_PRM_COLOR_FILTER_ARRAY or "cfa"

Description: Returns color filter array type of RAW data.
Type: Integer - Read only.
Values:

value	note
XI_CFA_NONE	Result pixels have no filters applied in this format
XI_CFA_BAYER_RGGB	
XI_CFA_CMYG	
XI_CFA_RGR	
XI_CFA_BAYER_BGGR	
XI_CFA_BAYER_GRBG	
XI_CFA_BAYER_GBRG	

Usage:

```
xiGetParamInt(handle, XI_PRM_COLOR_FILTER_ARRAY, &cfa);
```

XI_PRM_CMS or "cms"

Description: Enable or disable color management.
Type: Integer.
Default:XI_CMS_DIS
Possible Values:

value	note
XI_CMS_DIS	disables color management
XI_CMS_EN	enables color management (high CPU usage)
XI_CMS_EN_FAST	enables fast color management (high RAM usage)

Usage:

```
xiSetParamInt(handle, XI_PRM_CMS, XI_CMS_EN);
```

Note: This feature is in Beta stage.

XI_PRM_CMS_INTENT or "cms_intent"

Description: Defines rendering intents.
Type: Integer.
Default:XI_CMS_INTENT
Possible Values:

value	note
XI_CMS_INTENT_PERCEPTUAL	
XI_CMS_INTENT_RELATIVE_COLORIMETRIC	
XI_CMS_INTENT_SATURATION	
XI_CMS_INTENT_ABSOLUTE_COLORIMETRIC	

Absolute colorimetric

Absolute colorimetry and relative colorimetry actually use the same table but differ in the adjustment for the white point media. If the output device has a much larger gamut than the source profile, i.e., all the colors in the source can be represented in the output, using the absolute colorimetry rendering intent would "ideally" (ignoring noise, precision, etc.) give an exact output of the specified CIELAB values. Perceptually, the colors may appear incorrect, but instrument measurements of the resulting output would match the source. Colors outside of the proof print system's possible color are mapped to the boundary of the color gamut. Absolute colorimetry is useful to get an exact specified color (e.g., IBM blue), or to quantify the accuracy of mapping methods.

Relative colorimetric

The goal in relative colorimetry is to be truthful to the specified color, with only a correction for the media. Relative colorimetry is useful in proofing applications, since you are using it to get an idea of how a print on one device will appear on a different device. Media differences are the only thing you really would like to adjust for. Obviously there has to be some gamut mapping going on also. Usually this is done in a way where hue and lightness are maintained at the cost of reduced saturation. Relative colorimetric is the default rendering intent on most systems.

Perceptual and Saturation

The perceptual and saturation intents are where the results really depend upon the profile maker. This is even how some of the competitors in this market differentiate themselves. These intents should be created by the profile maker so that pleasing images occur with the perceptual intent while eye-catching business graphics occur with the saturation intent. This is achieved through the use of different perceptual remaps of the data as well as different gamut mapping methods. Perceptual rendering is recommended for color separation. In practice, photographers almost always use relative or perceptual intent, as for natural images, absolute causes color cast, while saturation produces unnatural colors.[6] Relative intent handles out-of-gamut by clipping (burning) these colors to the edge of the gamut, leaving in-gamut colors unchanged, while perceptual intent smoothly moves out-of-gamut colors into gamut, preserving gradations, but distorts in-gamut colors in the process. If an entire image is in-gamut, relative is perfect, but when there are out of gamut colors, which is more preferable depends on a case-by-case basis.

Saturation intent is most useful in charts and diagrams, where there is a discrete palette of colors that the designer wants saturated to make them intense, but where specific hue is less important.

Usage:

```
xiSetParamInt(handle, XI_PRM_CMS_INTENT, XI_CMS_INTENT_PERCEPTUAL);
```

Note: This feature is in Beta stage.

XI_PRM_APPLY_CMS or "apply_cms"

Description: If set to XI_ON applies CMS profile to xiGetImage.
Type: Integer.
Default:XI_OFF
Possible Values:
Usage:

```
xiSetParamInt(handle, XI_PRM_APPLY_CMS, XI_ON);
```

XI_PRM_INPUT_CMS_PROFILE or "input_cms_profile"

Description: Filename of the input cms profile (e.g. input.icc)
Type: String.
Default:Default device profile.
Usage:

```
xiSetParamString(h,XI_PRM_INPUT_CMS_PROFILE,"C:\\ICC\\custom_in.icc",(DWORD) strlen("C:\\ICC\\custom_in.icc"));
```

XI_PRM_OUTPUT_CMS_PROFILE or "output_cms_profile"

Description: Filename of the output cms profile (e.g. output.icc)
Type: String.
Default:(sRGB profile).
Usage:

```
xiSetParamString(h,XI_PRM_INPUT_CMS_PROFILE,"C:\\ICC\\custom_out.icc",(DWORD) strlen("C:\\ICC\\custom_out.icc"));
```

Device IO parameters.

XI_PRM_GPI_SELECTOR or “gpi_selector”

Description: Selects GPI.
Type: Integer.
Default: 1
Usage:

```
xiSetParamInt(handle, XI_PRM_GPI_SELECTOR, gpi_index);
```

Example: See XI_PRM_GPI_LEVEL

On each camera family or model the relation of gpi_index and physical pin differs. Following table list the camera families and gpi_index relations.

Families: xiQ/MQ

gpi_index	physical pin on the camera
1	1 (IN1 - optical)

Models: MC023,MT023,MC031,MT031,MC050,MT050,MC089,MT089,MC124,MT124

gpi_index	physical pin on the camera
1	5 (IN1 - optical)
2	8 (INOUT1)
3	2 (INOUT2)

Models: MX023,MX031,MX050,MX089,MX124

gpi_index	physical pin on the camera
1	22 (IN1 - optical)
2	20 (INOUT1)
3	21 (INOUT2)

Models: MX120,MX200

gpi_index	physical pin on the camera
1	2 (IN1 - optical)
2	4 (IN2 - optical)
3	6 (INOUT1)
4	7 (INOUT2)
5	45 (INOUT3)
6	46 (INOUT4)

Models: CB120,CB200

gpi_index	physical pin on the camera
1	3 (IN1 - optical)
2	4 (IN2 - optical)
3	6 (INOUT1)
4	7 (INOUT2)
5	11 (INOUT3)
6	12 (INOUT4)

Models: CB120-X8G3

gpi_index	physical pin on the camera
1	2 (IN1 - optical)
2	1 (IN2 - optical)
3	7 (INOUT1)
4	9 (INOUT2)
5	8 (INOUT3)
6	12 (INOUT4)

XI_PRM_GPI_MODE or “gpi_mode”

Description: Defines GPI functionality.
Type: Integer.
Default: XI_GPI_OFF
Possible values:

value	meaning
XI_GPI_OFF	Input is not used for triggering, but can be used to get XI_PRM_GPI_LEVEL. This can be used to switch I/O line on some cameras to input mode.
XI_GPI_TRIGGER	Input can be used for triggering
XI_GPI_EXT_EVENT	External signal input (not implemented)

Usage:

```
xiSetParamInt(handle, XI_PRM_GPI_MODE, XI_GPI_TRIGGER);
```

Note: To use GPI as trigger source, the XI_PRM_TRG_SOURCE should be also set to XI_TRG_EDGE_RISING or XI_TRG_EDGE_FALLING
Example:

```
// select digital input (for xiQ=1, for xiC=1 or 2)
int input_id = 1;
xiSetParamInt(handle, XI_PRM_GPI_SELECTOR,  input_id);
// set input as frame trigger
xiSetParamInt(handle, XI_PRM_GPI_MODE, XI_GPI_TRIGGER);
// enable triggering of image from digital input
xiSetParamInt(handle, XI_PRM_TRG_SOURCE, XI_TRG_EDGE_RISING);
```

Note: On CURRERA-R it is possible to test inputs using button connected to BOB144

XI_PRM_GPI_LEVEL or “gpi_level”

Description: Level of digital input selected by XI_PRM_GPI_SELECTOR.
Type: Integer.
Default: -
Note: When used on pin that could be input **or** output (E.g. pin 8 on MC023 camera), then associated GPO needs to be in mode XI_GPO_HIGH_IMPEDANCE. Otherwise pin can be pulled down (GPO_OFF) or up (GPO_ON). Such pins are HIGH_IMPEDANCE as default so application does not to setup it when used only as input.
Usage:

```
xiGetParamInt(handle, XI_PRM_GPI_LEVEL, &gpi_level);
```

Example:

```
// select digital input (different mapping to physical pin on each model, see table below)
int gpo_index = 1;
xiSetParamInt(handle, XI_PRM_GPI_SELECTOR,  gpo_index);
// get input level
int gpi_level = 0;
xiGetParamInt(handle, XI_PRM_GPI_LEVEL,  &gpi_level);
printf("Level on digital input %d is %d\n",pin,gpi_level);
```

Note: On CURRERA-R it is possible to test inputs using button connected to BOB144

XI_PRM_GPO_SELECTOR or “gpo_selector”

Description: Selects GPO.
Type: Integer.
Default: 1
Usage:

```
xiSetParamInt(handle, XI_PRM_GPO_SELECTOR, gpo_index);
```

Example: See XI_PRM_GPO_MODE
On each camera family or model the relation of gpo_index and physical pin differs. Following table list the camera families and gpi_index relations.
Families: xiQ/MQ

gpo_index	physical pin on the camera
1	2 (OUT1 - optical)

Models: MC023,MT023,MC031,MT031,MC050,MT050,MC089,MT089,MC124,MT124

gpo_index	physical pin on the camera
1	3 (OUT1 - optical)
2	8 (INOUT1)
3	2 (INOUT2)

Models: MX023,MX031,MX050,MX089,MX124

gpo_index	physical pin on the camera
1	24 (OUT1 - optical)
2	20 (INOUT1)
3	21 (INOUT2)

Models: MX120,MX200

gpo_index	physical pin on the camera
1	50 (OUT1 - optical)
2	48 (OUT2 - optical)
3	6 (INOUT1)
4	7 (INOUT2)
5	45 (INOUT3)
6	46 (INOUT4)

Models: CB120,CB200

gpo_index	physical pin on the camera
1	8 (OUT1 - optical)
2	9 (OUT2 - optical)
3	6 (INOUT1)
4	7 (INOUT2)
5	11 (INOUT3)
6	12 (INOUT4)

Models: CB120-X8G3

gpo_index	physical pin on the camera
1	4 (OUT1 - optical)
2	3 (OUT2 - optical)
3	7 (INOUT1)
4	9 (INOUT2)
5	8 (INOUT3)
6	12 (INOUT4)

XI_PRM_GPO_MODE or “gpo_mode”

Description: Defines GPO functionality.
Type: Integer.
Default: XI_GPO_OFF (or XI_GPO_HIGH_IMPEDANCE when INOUTx pin is selected)
Possible values:

value	meaning
XI_GPO_OFF**	Output is **off** (zero voltage or switched_off)
XI_GPO_ON**	Output is **on** (voltage or switched_on)
XI_GPO_FRAME_ACTIVE*	Output is **on** while frame exposure,read,transfer
XI_GPO_FRAME_ACTIVE_NEG*	Output is **off** while frame exposure,read,transfer
XI_GPO_EXPOSURE_ACTIVE*	Output is **on** while frame exposure
XI_GPO_EXPOSURE_ACTIVE_NEG*	Output is **off** while frame exposure
XI_GPO_FRAME_TRIGGER_WAIT*	Output is **on** while camera is ready for trigger
XI_GPO_FRAME_TRIGGER_WAIT_NEG*	Output is **off** while camera is ready for trigger
XI_GPO_EXPOSURE_PULSE***	Output is **on** short pulse at the beginning of frame exposure
XI_GPO_EXPOSURE_PULSE_NEG***	Output is **off** short pulse at the beginning of frame exposure
XI_GPO_BUSY	Output is **on** when camera has received trigger until end of transfer
XI_GPO_BUSY_NEG	Output is **off** when camera has received trigger until end of transfer
XI_GPO_HIGH_IMPEDANCE	Associated pin is in high impedance (tri-stated) and can be driven externally. E.g. for triggering or reading status by GPI_LEVEL.

*Note1: On some camera models (MR, MH): Modes FRAME_ACTIVE or EXPOSURE_ACTIVE are supported only if XI_PRM_TRG_SOURCE is set to XI_TRG_SOFTWARE or XI_TRG_EDGE_RISING or XI_TRG_EDGE_FALLING. See section TriggerSource
On models xiMU (MU9), xiQ, CURRERA-R the XI_GPO_EXPOSURE_ACTIVE can be used when XI_PRM_TRG_SOURCE is OFF (Live mode).
**Note2: Some camera families (e.g. MR) does not support the software control of outputs. Only one of mode: FRAME_ACTIVE and EXPOSURE_ACTIVE can be set.
***Note3: Duration of pulse depends on camera model and polarity of signal.
****Note4: Each bidirectional line has only one control for inverter (as in GenICam-SFNC). If output mode with _NEG extension is set then also input signal becomes inverted.
Usage:

```
xiSetParamInt(handle, XI_PRM_GPO_MODE, XI_GPO_ON);
```

Example:

```
// select digital output (on CURRERA-R 1-4)
xiSetParamInt(handle, XI_PRM_GPO_SELECTOR, 1);
// make output on (one)
xiSetParamInt(handle, XI_PRM_GPO_MODE,  XI_GPO_ON);
Sleep(1000); // wait to see the output is on (e.g. LED)
// make output off
xiSetParamInt(handle, XI_PRM_GPO_MODE,  XI_GPO_OFF);
```

Note: On CURRERA-R it is possible to test output state using LED connected to BOB144

XI_PRM_LED_SELECTOR or “led_selector”

Description: Selects LED.
Type: Integer.
Default: 1
Usage:

```
xiSetParamInt(handle, XI_PRM_LED_SELECTOR, 1);
```

XI_PRM_LED_MODE or “led_mode”

Description: Defines LED functionality.
Type: Integer.
Default: -
Possible values:

value	meaning
XI_LED_HEARTBEAT	Set led to blink (1 Hz) if link is OK.
XI_LED_TRIGGER_ACTIVE	Set led to blink if trigger detected.
XI_LED_EXT_EVENT_ACTIVE	Set led to blink if external signal detected.
XI_LED_ACQUISITION	Set led to blink if data streaming.
XI_LED_EXPOSURE_ACTIVE	Set led to blink if sensor integration time.
XI_LED_FRAME_ACTIVE	Set led to blink if device busy/not busy.
XI_LED_LINK	Set led to blink if link is ok.
XI_LED_OFF	Set led to off.
XI_LED_ON	Set led to on.

Usage:

```
xiSetParamInt(handle, XI_PRM_LED_MODE, XI_LED_TRIGGER_ACTIVE);
```

XI_PRM_TS_RST_SOURCE “ts_rst_source”

Description: Defines source for time stamp reset engine as well as the polarity active signal. Engine is edge sensitive.
Type: Integer.
Default: XI_TS_RST_OFF
Possible values:

value	meaning
XI_TS_RST_OFF	No source selected time stamp reset is not armed
XI_TS_RST_SRC_GPI_1	GPI1 rising edge is active (signal after de-bounce module)
XI_TS_RST_SRC_GPI_2	GPI2 rising edge is active
XI_TS_RST_SRC_GPI_3	GPI3 rising edge is active
XI_TS_RST_SRC_GPI_4	GPI4 rising edge is active
XI_TS_RST_SRC_GPI_1_INV	GPI1 falling edge is active
XI_TS_RST_SRC_GPI_2_INV	GPI2 falling edge is active
XI_TS_RST_SRC_GPI_3_INV	GPI3 falling edge is active
XI_TS_RST_SRC_GPI_4_INV	GPI4 falling edge is active
XI_TS_RST_SRC_GPO_1	GPO1 rising edge is active (signal after de-bounce module)
XI_TS_RST_SRC_GPO_2	GPO2 rising edge is active
XI_TS_RST_SRC_GPO_3	GPO3 rising edge is active
XI_TS_RST_SRC_GPO_4	GPO4 rising edge is active
XI_TS_RST_SRC_GPO_1_INV	GPO1 falling edge is active
XI_TS_RST_SRC_GPO_2_INV	GPO2 falling edge is active
XI_TS_RST_SRC_GPO_3_INV	GPO3 falling edge is active
XI_TS_RST_SRC_GPO_4_INV	GPO4 falling edge is active
XI_TS_RST_SRC_TRIGGER	TRIGGER to sensor rising edge is active
XI_TS_RST_SRC_TRIGGER_INV	TRIGGER to sensor falling edge is active
XI_TS_RST_SRC_SW	time stamp is reset by software take effect imminently
XI_TS_RST_SRC_EXPACTIVE	Exposure Active signal rising edge
XI_TS_RST_SRC_EXPACTIVE_INV	Exposure Active signal falling edge
XI_TS_RST_SRC_FVAL	Frame valid signal rising edge (internal signal in camera)
XI_TS_RST_SRC_FVAL_INV	Frame valid signal falling edge (internal signal in camera)

Note: Number of active GPI or GPO depends on camera model
Usage:

```
xiSetParamInt(handle, XI_PRM_TS_RST_MODE , XI_TS_RST_ARM_ONCE);
xiSetParamInt(handle, XI_PRM_TS_RST_SOURCE , XI_TS_RST_SRC_TRIGGER);
```

XI_PRM_TS_RST_MODE “ts_rst_mode”

Description: Defines way time stamp reset engine is armed.
Type: Integer.
Default: XI_TS_RST_ARM_ONCE
Possible values:

value	meaning
XI_TS_RST_ARM_ONCE	Engine is disabled after time stamp has been reset after selected event.
XI_TS_RST_ARM_PERSIST	Engine is armed permanently so each selected event will trigger time stamp reset.

Usage:

```
xiSetParamInt(handle, XI_PRM_TS_RST_MODE , XI_TS_RST_ARM_ONCE);
xiSetParamInt(handle, XI_PRM_TS_RST_SOURCE , XI_TS_RST_SRC_TRIGGER);
```

Extended Device Parameters.

XI_PRM_ACQ_BUFFER_SIZE or “acq_buffer_size”

Description: Defines the size of the acquisition buffer in bytes(see Image below). This is a circle buffer which contains image data from sensor. This parameter can be set only when acquisition is stopped.
Type: Integer.
Defaults:
Usage:

```
xiSetParamInt(handle, XI_PRM_ACQ_BUFFER_SIZE, buf_size_bytes);
```

Example:

- Application sets buffer size to 70MB (on CURRERA-R)

```
xiSetParamInt(handle, XI_PRM_ACQ_BUFFER_SIZE, 70*1000*1000)
```

- Sensor gives 1MB of data per image @ 10 frames/second
- Application retrieves image data by xiGetImage
- Application has access to the frame at most 7 seconds (70MB/(1MB*10fps))
  Note: If the processing of this image takes more time than these 7seconds, the image data will be automatically overwritten with new image data due to the circular character of the buffer.

XI_PRM_ACQ_BUFFER_SIZE_UNIT or "acq_buffer_size_unit"

Description: Acquisition buffer size unit. Default 1. E.g. Value 1024 means that buffer_size is in KiBytes.
Type: Integer.
Default: 1
Usage:

```
// set unit to 1 MiB
xiSetParamInt(handle, XI_PRM_ACQ_BUFFER_SIZE_UNIT, 1024*1024);
// set buffer size to 200 MiB
xiSetParamInt(handle, XI_PRM_ACQ_BUFFER_SIZE, 200);
```

XI_PRM_ACQ_TRANSPORT_BUFFER_SIZE or "acq_transport_buffer_size"

Description: Size of one transport buffer in bytes. Frame/Field can contain multiple transport buffers. To decrease CPU load and increase system performance on committing transport buffers to kernel driver, transport buffer size has to be as high as possible. However in case of small Frame/Field size and high framerates it is necessary to decrease transport buffer size and increase queue of Frame/Field buffers(XI_PRM_BUFFERS_QUEUE_SIZE). Check out How to optimize software performance on high frame rates for more info.
Type: Integer.
Default: Value compatible for all tested controllers.
Note: Whole range minimum to maximum is not guaranteed on all tested configurations. Please be aware of possible issues on some controllers.
Usage:

```
xiSetParamInt(handle, XI_PRM_ACQ_TRANSPORT_BUFFER_SIZE, size_in_bytes);
```

Example:

- Application set transport buffer size to 128KiB for small (80KB images)

```
xiSetParamInt(handle, XI_PRM_ACQ_TRANSPORT_BUFFER_SIZE, 128*1024);
```

XI_PRM_ACQ_TRANSPORT_PACKET_SIZE or "acq_transport_packet_size"

Description: Acquisition transport packet size in bytes.
Type: Integer.
Default: by interface
Usage:

```
// Get packet size
xiGetParamInt(handle, XI_PRM_ACQ_TRANSPORT_PACKET_SIZE, &packet_size);
```

XI_PRM_BUFFERS_QUEUE_SIZE or “buffers_queue_size”

Description: *XI_PRM_BUFFERS_QUEUE_SIZE - 1* is the maximum number of images which can be stored in the buffers queue.
Type: Integer.
Default: 4 (Which means 3 images can be stored in the queue)
Invalidators: XI_PRM_ACQ_BUFFER_SIZE, XI_PRM_ACQ_BUFFER_SIZE_UNIT

Usage:

```
xiSetParamInt(handle, XI_PRM_BUFFERS_QUEUE_SIZE, images_count);
```

Example:

- Application sets queue size to 7

```
xiSetParamInt(handle, XI_PRM_BUFFERS_QUEUE_SIZE, 7);
xiSetParamInt(handle, XI_PRM_ACQ_BUFFER_SIZE, 40*1024*1024);
```

- Sensor acquires images at 10 frames/second. It means new image is acquired every 100ms, therefore queue can contain up to 6 images acquired during 600ms (=6*100ms) time span.
- Application needs typically 40ms for processing of an image.
- If it takes 400ms for the application to process one of the images (e.g. saving it to a server) the subsequent 4 frames are still available since they are stored in the queue (images 17-19 on the below figure).

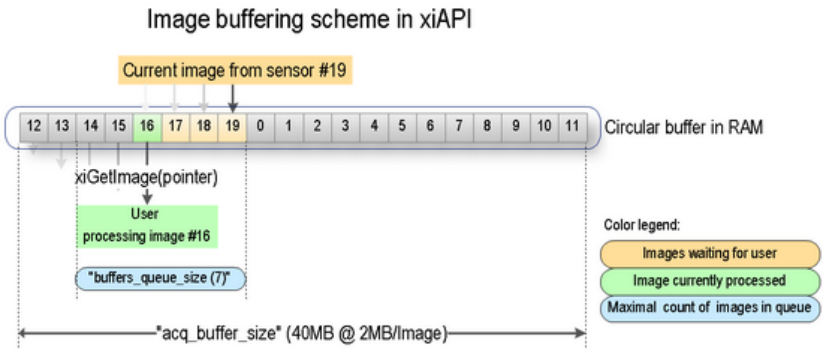


Illustration of normal status when application is processing one frame longer

- If it takes 1000ms for the application to process one of the images, the buffer queue will fill up and some images will be skipped (images 16-20 on the below figure). The user can check for such situation by checking the image sequence number (nframe of XI_IMG structure) of each image.

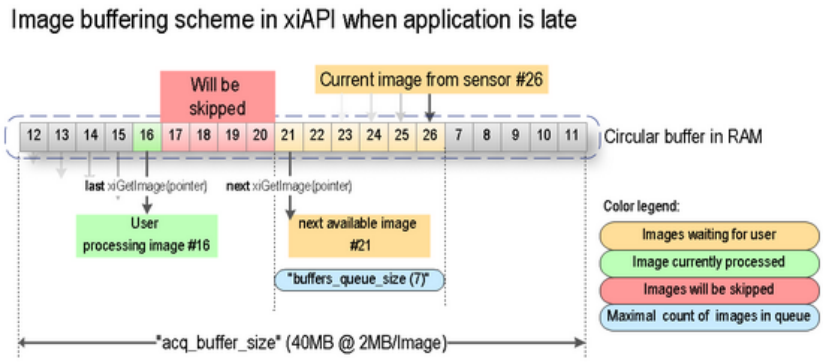


Illustration of state when application gets late. Longer than queue.

- The application can work with the image data at most **acq_buffer_size/(frame_rate_FPS*bytes_per_frame)** seconds before the data is overwritten due to the circular character of the buffer. If the application needs more time to process the image, XI_PRM_BUFFER_POLICY must be set to XI_BP_SAFE. In this case API copies the image to a user/API allocated memory where it can be accessed without the risk of being overwritten. This copying however takes extra CPU time.

XI_PRM_ACQ_TRANSPORT_BUFFER_COMMIT or "acq_transport_buffer_commit"

Description: Defines number of buffers to commit to low level.
Type: Integer.
Defaults:
Usage:

```
xiSetParamInt(handle, XI_PRM_ACQ_TRANSPORT_BUFFER_COMMIT, number_of_buffers);
```

XI_PRM_RECENT_FRAME or "recent_frame"

Description: This parameter changes the behavior of xiGetImage.

value	xiGetImage behavior
0 (default)	Retrieves next available image from buffer
1	Retrieves the most recent image from buffer

Type: Integer.
Default: 0 (=off)
Usage:

```
xiSetParamInt(handle, XI_PRM_RECENT_FRAME, 1);
```

XI_PRM_TRANSPORT_DATA_TARGET or “transport_data_target”

Description: Sets image data delivery target to CPU RAM (default) or GPU RAM.
Type: Integer.
Default: 0 (XI_TRANSPORT_DATA_TARGET_CPU_RAM).

target	value	comment
XI_TRANSPORT_DATA_TARGET_CPU_RAM	0	normal CPU memory buffer is used for image data
XI_TRANSPORT_DATA_TARGET_GPU_RAM	1	data is delivered straight to GPU memory using GPUDirect technology: see detailed description

Usage:

```
xiSetParamInt(handle, XI_PRM_TRANSPORT_DATA_TARGET, XI_TRANSPORT_DATA_TARGET_GPU_RAM);
```

XI_PRM_DEVICE_RESET or “device_reset”

Description: Resets the camera firmware. From functional view it is the same like disconnection and connection of the camera.
It is typically followed by enumeration of operating system which might take some time (e.g. 10 seconds).
Application shall wait some time after the reset and then use xiGetNumberDevices in order to enumerate the camera again.
Type: Integer.
Default: 0 (disabled)
Note: currently supported only for xiQ camera family
Usage:

```
xiSetParamInt(handle, XI_PRM_DEVICE_RESET, 1);
Sleep(10000); // wait 10 seconds for enumeration by OS
DWORD devices_count=0;
xiGetNumberDevices(&devices_count);
```

XI_PRM_AEAG or “aeag”

Description: Automatic exposure/gain.
Type: Integer.
Default: 0 (disabled)
Usage:

```
xiSetParamInt(handle, XI_PRM_AEAG, 1);
```

XI_PRM_AE_MAX_LIMIT or “ae_max_limit”

Description: Maximum limit of exposure (in uSec) in AEAG procedure.
Type: Integer.
Default: 100000 uSec
Usage:

```
xiSetParamFloat(handle, XI_PRM_AE_MAX_LIMIT, 123);
```

XI_PRM_AG_MAX_LIMIT or “ag_max_limit”

Description: Maximum limit of gain in AEAG procedure.
Type: Float.
Default: depends on camera type (dB)
Usage:

```
xiSetParamFloat(handle, XI_PRM_AG_MAX_LIMIT, 12);
```

XI_PRM_EXP_PRIORITY or “exp_priority”

Description: Exposure priority for Auto Exposure / Auto Gain function.

Value	Meaning
1	Exposure priority. Only exposure will be changed.
0.5	Exposure and gain will be used (50%:50%)
0	Gain priority. Only gain will be changed.

Type: Float.
Default: 0.8
Usage:

```
xiSetParamFloat(handle, XI_PRM_EXP_PRIORITY, 0.9);
```

XI_PRM_AEAG_LEVEL or “aeag_level”

Description: Average intensity of output signal AEAG should achieve(in %).
Type: Float.
Default: 40 (%)
Usage:

```
xiSetParamFloat(handle, XI_PRM_AEAG_LEVEL, 50);
```

XI_PRM_AEAG_ROI_OFFSET_X or “aeag_roi_offset_x”

Description: X offset of the area used for AEAG calculation. The sum of XI_PRM_AEAG_ROI_OFFSET_X and XI_PRM_AEAG_ROI_WIDTH must be equal or lower than the image resolution(width).
Type: Int.
Default: 0
Usage:

```
xiSetParamInt(handle, XI_PRM_AEAG_ROI_OFFSET_X,40);
```

XI_PRM_AEAG_ROI_OFFSET_Y or “aeag_roi_offset_y”

Description: Y offset of the area used for AEAG calculation. The sum of XI_PRM_AEAG_ROI_OFFSET_Y and XI_PRM_AEAG_ROI_HEIGHT must be equal or lower than the image resolution(height).
Type: Int.
Default: 0
Usage:

```
xiSetParamInt(handle, XI_PRM_AEAG_ROI_OFFSET_Y,40);
```

XI_PRM_AEAG_ROI_WIDTH or “aeag_roi_width”

Description: width of the area used for AEAG calculation. The sum of XI_PRM_AEAG_ROI_OFFSET_X and XI_PRM_AEAG_ROI_WIDTH must be equal or lower than the image resolution(width).
Type: Int.
Default: depends on the sensors resolution and downsampling
Usage:

```
xiSetParamInt(handle, XI_PRM_AEAG_ROI_WIDTH,400);
```

XI_PRM_AEAG_ROI_HEIGHT or “aeag_roi_height”

Description: height of the area used for AEAG calculation. The sum of XI_PRM_AEAG_ROI_OFFSET_Y and XI_PRM_AEAG_ROI_HEIGHT must be equal or lower than the image resolution(height).
Type: Int.
Default: depends on the sensors resolution and downsampling
Usage:

```
xiSetParamInt(handle, XI_PRM_AEAG_ROI_HEIGHT,400);
```

XI_PRM_DEBOUNCE_EN or “dbnc_en”

Description: Enable/Disable debounce to selected GPI (see XI_PRM_GPI_SELECTOR parameter).
Type: Integer.
Default: 0 (disabled)
Usage:

```
xiSetParamInt(handle, XI_PRM_DEBOUNCE_EN, 1);
```

XI_PRM_DEBOUNCE_T0 or “dbnc_t0”

Description: Debounce time (x * 10us) for transition to inactive level of GPI selected by XI_PRM_DEBOUNCE_POL(see XI_PRM_DEBOUNCE_POL).
Type: Integer.
Default: 0
Usage:

```
xiSetParamInt(handle, XI_PRM_DEBOUNCE_T0 , 10);
```

XI_PRM_DEBOUNCE_T1 or “dbnc_t1”

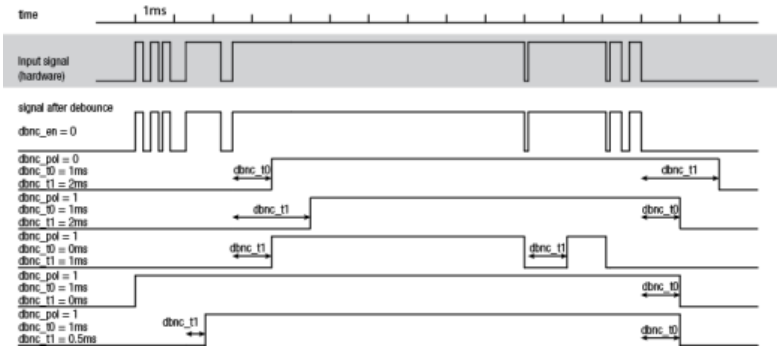
Description: Debounce time (x * 10us)for transition to active level of GPI selected by XI_PRM_DEBOUNCE_POL(see XI_PRM_DEBOUNCE_POL).
Type: Integer.
Default: 0
Usage:

```
xiSetParamInt(handle, XI_PRM_DEBOUNCE_T1 , 10);
```

XI_PRM_DEBOUNCE_POL or “dbnc_pol”

Description: Debounce polarity selects active level of GPI (see XI_PRM_GPI_SELECTOR parameter). Does not inverts the signal if set.
Type: Integer.
Default: 1 (disabled)
Usage:

```
xiSetParamInt(handle, XI_PRM_DEBOUNCE_POL, 1);
```



Temperature

XI_PRM_IS_COOLED or "iscooled"

Description: Returns 1 for cameras that support cooling.
Type: Integer. Read only.
Default: By default -.
Usage:

```
xiGetParamInt(handle, XI_PRM_IS_COOLED, &is_cooled);
```

XI_PRM_COOLING or "cooling"

Description: Set camera cooling control. Replaced by XI_PRM_TEMP_CONTROL_MODE
Type: Enumerator.
Default: By default 0 (Off).

Value	Mode
XI_TEMP_CTRL_MODE_OFF	Controlling of elements (TEC/Peltier, Fans) is turned off
XI_TEMP_CTRL_MODE_AUTO	Controlling of elements is performed automatically by API or camera in order to reach XI_PRM_TARGET_TEMP
XI_TEMP_CTRL_MODE_MANUAL	Controlling of elements is done manually by application

Usage:

```
xiSetParamInt(handle, XI_PRM_TEMP_SELECTOR, XI_TEMP_SENSOR_BOARD);
xiSetParamFloat(handle, XI_PRM_TARGET_TEMP, 18.5);
xiSetParamInt(handle, XI_PRM_COOLING, XI_TEMP_CTRL_MODE_AUTO);
```

XI_PRM_TARGET_TEMP or "target_temp"

Description: Set target temperature for temperature control. XI_PRM_TEMP_SELECTOR might be used for selection of thermometer.
Type: Float.
Default: By default 20 °C.
Usage:

```
xiSetParamFloat(handle, XI_PRM_TARGET_TEMP, 18.5);
xiGetParamFloat(handle, XI_PRM_TARGET_TEMP, &target_temp);
```

XI_PRM_TEMP_SELECTOR or "temp_selector"

Description: Temperature sensor selector.
Type: Integer.
Default: 0 (XI_TEMP_IMAGE_SENSOR_RAW)
The parameter selects one of temperature sensors.

Value	Thermometer location
XI_TEMP_IMAGE_SENSOR_RAW	Image sensor die (non-calibrated)

XI_TEMP_IMAGE_SENSOR	Image sensor die (calibrated)
XI_TEMP_SENSOR_BOARD	PCB as image sensor
XI_TEMP_INTERFACE_BOARD	Interface board
XI_TEMP_FRONT_HOUSING	Front part of camera housing
XI_TEMP_BACK_HOUSING	Rear part of camera housing

Usage:

```
float temperature=0;
xiSetParamInt(handle, XI_PRM_TEMP_SELECTOR, XI_TEMP_IMAGE_SENSOR_RAW);
xiGetParamFloat(handle, XI_PRM_TEMP, &temperature);
```

XI_PRM_TEMP or "temp"

Description: Selected thermometer reading in degree Celsius. Thermometer can be selected by XI_PRM_TEMP_SELECTOR.
Type: Float.
Default: By default -.
Usage:

```
float temperature_c=0;
xiGetParamFloat(handle, XI_PRM_TEMP, &temperature_c);
```

XI_PRM_CHIP_TEMP or "chip_temp"

Description: Temperature reading of thermometer chip. Sensor is located on the PCB close to imaging sensor. Units: degrees of Celsius.
Type: Float.
Default: By default -.
Usage:

```
xiGetParamFloat(handle, XI_PRM_CHIP_TEMP, &temperature_chip_temp);
```

XI_PRM_HOUS_TEMP or "hous_temp"

Description: Camera housing temperature.
Type: Float.
Default: By default -.
Usage:

```
xiGetParamFloat(handle, XI_PRM_HOUS_TEMP, &house_temp);
```

XI_PRM_HOUS_BACK_SIDE_TEMP or "hous_back_side_temp"

Description: Camera housing back side temperature.
Type: Float.
Default: By default -.
Usage:

```
xiGetParamFloat(handle, XI_PRM_HOUS_BACK_SIDE_TEMP, &house_temp);
```

XI_PRM_SENSOR_BOARD_TEMP or "sensor_board_temp"

Description: Camera sensor board temperature.
Type: Float.
Default: By default -.
Usage:

```
xiGetParamFloat(handle, XI_PRM_SENSOR_BOARD_TEMP, &sensor_board_temp);
```

XI_PRM_TEMP_CONTROL_MODE or "device_temperature_ctrl_mode"

Description: Temperature control mode
Type: Enumerator.
Default: 0 (XI_TEMP_CTRL_MODE_OFF)
The parameter selects one of temperature sensors.

Value	Mode
XI_TEMP_CTRL_MODE_OFF	Controlling of elements (TEC/Peltier, Fans) is turned off
XI_TEMP_CTRL_MODE_AUTO	Controlling of elements is performed automatically by API or camera in order to reach XI_PRM_TARGET_TEMP
XI_TEMP_CTRL_MODE_MANUAL	Controlling of elements is done manually by application

Usage:

```
xiSetParamInt(handle, XI_PRM_TEMP_SELECTOR, XI_TEMP_SENSOR_BOARD);
xiSetParamFloat(handle, XI_PRM_TARGET_TEMP, 18.5);
xiSetParamInt(handle, XI_PRM_COOLING, XI_TEMP_CTRL_MODE_AUTO);
```

XI_PRM_TEMP_ELEMENT_SEL or "device_temperature_element_sel"

Description: Temperature element selector (TEC, Fan)
Type: Enumerator.
Default: 0 (XI_TEMP_ELEM_TEC1)

Value	Element
XI_TEMP_ELEM_TEC1	TEC1 = TEC/Peltier that is closest to the image sensor
XI_TEMP_ELEM_TEC2	TEC2 = TEC/Peltier location depends on camera model
XI_TEMP_ELEM_FAN1	FAN1 = Fan

Usage: See XI_PRM_TEMP_ELEMENT_VALUE

XI_PRM_TEMP_ELEMENT_VALUE or "device_temperature_element_val"

Description: Temperature element value in percents of full control range
Type: Float.
Default: By default -.
Usage:

```
xiSetParamInt(handle, XI_PRM_COOLING, XI_TEMP_CTRL_MODE_MANUAL);
xiSetParamInt(handle, XI_PRM_TEMP_ELEMENT_SEL, XI_TEMP_ELEM_TEC1);
xiSetParamFloat(handle, XI_PRM_TEMP_ELEMENT_VALUE, 50.1);
```

Device Connection

XI_PRM_IS_DEVICE_EXIST or "isexist"

Description: Returns 1 if camera connected and works properly.
Type: Integer.
Usage:

```
xiGetParamInt(handle, XI_PRM_IS_DEVICE_EXIST, &is_ok);
```

Sensor Defects Correction parameters

XI_PRM_SENS_DEFECTS_CORR_LIST_SELECTOR “bpc_list_selector”

Description: Selector for current sensor defects list used by Sensor Defect Correction.
Type: Enumerator.
Default: By default 0 (Off).

Value	Mode
XI_SENS_DEFFECTS_CORR_LIST_SEL_FACTORY	List calibrated in camera production factory
XI_SENS_DEFFECTS_CORR_LIST_SEL_USER0	User list, created by user

Usage:

```
xiSetParamInt(handle, XI_PRM_SENS_DEFECTS_CORR_LIST_SELECTOR , XI_SENS_DEFFECTS_CORR_LIST_SEL_USER0 );
xiSetParamInt(handle, XI_PRM_SENS_DEFECTS_CORR, XI_ON);
```

XI_PRM_SENS_DEFECTS_CORR_LIST_CONTENT “sens_defects_corr_list_content”

Description: Set/Get current sensor defects list used by Sensor Defect Correction(in specific text format).
Type: String.
Default: By default -
Usage:

```
xiSetParamString(handle, XI_PRM_SENS_DEFECTS_CORR_LIST_CONTENT , string, strlen(string) );
xiGetParamString(handle, XI_PRM_SENS_DEFECTS_CORR_LIST_CONTENT, string, string_size);
```

XI_PRM_SENS_DEFECTS_CORR or “bpc”

Description: Correction of sensor defects.
Type: Integer.
Default: 0 (disabled)
Usage:

```
xiSetParamInt(handle, XI_PRM_SENS_DEFECTS_CORR, 1);
```

XI_PRM_FFC or “ffc”

Description: Image flat field correction.(XI_PRM_NEW_PROCESS_CHAIN_ENABLE must be XI_ON).
For more information see Flat Field Correction support page.
Type: Integer.
Default: 0 (disabled)
Usage:

```
xiSetParamInt(handle, XI_PRM_FFC, 1);
```

XI_PRM_FFC_FLAT_FIELD_FILE_NAME or "ffc_flat_field_file_name"

Description: Set name of file of image flat field to be applied for FFC processor(in tiff format). (XI_PRM_NEW_PROCESS_CHAIN_ENABLE must be XI_ON).
For more information see Flat Field Correction support page.
Type: String.
Default: By default -.
Usage:

```
xiSetParamString(handle, XI_PRM_FFC_FLAT_FIELD_FILE_NAME, filename, size);
```

XI_PRM_FFC_DARK_FIELD_FILE_NAME or "ffc_dark_field_file_name"

Description: Set name of file of image dark field to be applied for FFC processor(in tiff format). (XI_PRM_NEW_PROCESS_CHAIN_ENABLE must be XI_ON).
For more information see Flat Field Correction support page.
Type: String.
Default: By default -.
Usage:

```
xiSetParamString(handle, XI_PRM_FFC_DARK_FIELD_FILE_NAME, filename, size);
```

XI_PRM_COLUMN_FPN_CORRECTION or “column_fpn_correction”

Description: Correction of column fpn.
Type: Integer.
Default: 0 (disabled)
Usage:

```
xiSetParamInt(handle, XI_PRM_COLUMN_FPN_CORRECTION, 1);
```

XI_PRM_ROW_FPN_CORRECTION or “row_fpn_correction”

Description: Correction of row fpn.
Type: Integer.
Default: 0 (disabled)
Usage:

```
xiSetParamInt(handle, XI_PRM_ROW_FPN_CORRECTION, 1);
```

Sensor Specific Parameters

XI_PRM_HDR or "hdr"

Note: ebables HDR mode for certain type of sensors. For more information see HDR mode support page.
Description: Enable High Dynamic Range sensor feature.
Type: Integer.
Default: By default 0.
Usage:

```
xiSetParamInt(handle, XI_PRM_HDR, 1);
xiGetParamInt(handle, XI_PRM_HDR, &hdr_enabled);
```

XI_PRM_HDR_KNEEPOINT_COUNT or "hdr_kneepoint_count"

Note: Defines the number of kneepoints in the PWLR curve.
Description: number of kneepoints.
Type: Integer.
Default: depends on sensor.
Usage:

```
xiSetParamInt(handle, XI_PRM_HDR_KNEEPOINT_COUNT, 1);
xiGetParamInt(handle, XI_PRM_HDR_KNEEPOINT_COUNT, &hdr_kp_count);
```

XI_PRM_HDR_T1 or "hdr_t1"

Description: Exposure time of first slope(in % of XI_PRM_EXPOSURE)
Type: Integer.
Default: By default 60%
Usage:

```
xiSetParamInt(handle, XI_PRM_HDR_T1,70);
```

XI_PRM_HDR_T2 or "hdr_t2"

Description: Exposure time of second slope(in % of XI_PRM_EXPOSURE)
Type: Integer.
Default: By default 30%
Usage:

```
xiSetParamInt(handle, XI_PRM_HDR_T2,20);
```

XI_PRM_KNEEPOINT1 or "hdr_kneepoint1"

Description: First kneepoint (% of sensor saturation)
Type: Integer.
Default: By default 40
Usage:

```
xiSetParamInt(handle, XI_PRM_KNEEPOINT1, 30);
```

XI_PRM_KNEEPOINT2 or "hdr_kneepoint2"

Description: Second kneepoint (% of sensor saturation)
Type: Integer.
Default: By default 60
Usage:

```
xiSetParamInt(handle, XI_PRM_KNEEPOINT2, 70);
```

XI_PRM_SENSOR_MODE or "sensor_mode"

Description: For some cameras this sensor specific parameter allows to select from predefined sensor modes. Setting of this parameter affects the image dimensions and downsampling.
Type: Integer.
Default: 0
Usage:

```
xiSetParamInt(handle, XI_PRM_SENSOR_MODE, 1);
```

XI_PRM_IMAGE_BLACK_LEVEL or "image_black_level"

Description: Black level is calculated level (in pixel counts) that should reflect the value of pixels without light. It should be the same as XI_IMG.black_level from last image get using xiGetImage.
Type: Integer. Read Only.
Default: 0

XI_PRM_API_CONTEXT_LIST or "xiapi_context_list"

Description: API Context contains text representation of current settings for offline image processing. It can be get while acquisition to store the context. Respectively it can be set while offline processing - to restore context.
Type: Text.

Api version and device driver/firmware info

XI_PRM_API_VERSION or "api_version"

Description: Returns the version of API.
Type: String.
Default: By default -.
Usage:

```
xiGetParamString(handle, XI_PRM_API_VERSION, char *, size);
```

XI_PRM_DRV_VERSION or "drv_version"

Description: Returns the version of the current device driver.
Type: String.
Default: By default -.
Usage:

```
xiGetParamString(handle, XI_PRM_DRV_VERSION, char *, size);
```

XI_PRM_MCU1_VERSION or "version_mcu1"

Description: Returns the version of the current MCU1 firmware.
Type: String.
Default: By default -.
Usage:

```
xiGetParamString(handle, XI_PRM_MCU1_VERSION, char *, size);
```

XI_PRM_MCU2_VERSION or "version_mcu2"

Description: Returns the version of the current MCU2 firmware.
Type: String.
Default: By default -.
Usage:

```
xiGetParamString(handle, XI_PRM_MCU2_VERSION, char *, size);
```

XI_PRM_FPGA1_VERSION or "version_fpga1"

Description: Returns the version of the current FPGA1 firmware.
Type: String.
Default: By default -.
Usage:

```
xiGetParamString(handle, XI_PRM_FPGA1_VERSION, char *, size);
```

XI_PRM_XMLMAN_VERSION or "version_xmlman"

Description: Returns version of XML manifest.
Type: String.
Default: By default -.
Usage:

```
xiGetParamString(handle, XI_PRM_XMLMAN_VERSION, char *, size);
```

XI_PRM_HW_REVISION or "hw_revision"

Description: Returns the hardware revision number of the camera.
Type: String.
Default: By default -.
Usage:

```
xiGetParamInt(handle, XI_PRM_HW_REVISION, int *);
```

Flash file system control

Some of XIMEA cameras contain Flash File System. It allows to store/read small customer file in each camera.
Example program for working with FFS

XI_PRM_FFS_FILE_NAME or "ffs_file_name"

Description: Name of file to be written/read from camera FFS.
Type: String.
Default: By default -.
Usage:

```
xiSetParamString(handle, XI_PRM_FFS_FILE_NAME, filename, size);
```

XI_PRM_READ_FILE_FFS or "read_file_ffs"

Description: File data to be read from camera flash file system.
Type: String. Read only.
Default: By default -.
Usage:

```
xiGetParamString(handle, XI_PRM_READ_FILE_FFS, data_buff, max_size);
```

Example: Example below reads the sensor defects file. This file exist on xiQ cameras.

```
// set filename
char filename[100]="bad_pixel_list.txt";
stat = xiSetParamString(xiH, XI_PRM_FFS_FILE_NAME, filename, sizeof(filename));
HandleResult(stat,"xiSetParamString (XI_PRM_FFS_FILE_NAME)");
// allocate buffer
#define MAX_FILE_SIZE 1000*1000 // 1MB
char* file_content=NULL;
file_content = (char*) calloc(1,MAX_FILE_SIZE);
if (!file_content)
{
  printf("Error on memory allocation for file content.\n");
  return;
}
// read file
stat = xiGetParamString(xiH, XI_PRM_READ_FILE_FFS, file_content, MAX_FILE_SIZE);
HandleResult(stat,"xiGetParamString (XI_PRM_READ_FILE_FFS)");
// print file content
printf("Text read from FFS file:%s\n%s\n\n", filename, file_content);
free(file_content);
```

HSI Calibration Files Access

For reading out Hyper-Spectral sensor calibration data from the camera - use the same code as above, but replace the filename with 'sens_calib.dat'
Code to be used for HSI Calibration:

```
// set filename for HSI Sensor Calibration
char filename[100]="sens_calib.dat";
stat = xiSetParamString(xiH, XI_PRM_FFS_FILE_NAME, filename, sizeof(filename));
// continue like in example for reading FFS file
```

XI_PRM_WRITE_FILE_FFS or "write_file_ffs"

Description: File data to be written to camera flash file system.
Type: String. Write only.
Default: By default -.
Usage:

```
xiSetParamString(handle, XI_PRM_WRITE_FILE_FFS, data_buff, size);
```

Delete file:

```
xiSetParamString(xiH, XI_PRM_FFS_FILE_NAME, "filename.txt", strlen("filename.txt"));
xiSetParamString(xiH, XI_PRM_WRITE_FILE_FFS, NULL, 0);
```

XI_PRM_FFS_FILE_ID or "ffs_file_id"

Description: File number(id) in camera FFS.
Type: Integer.
Default: By default 0.
Usage:

```
xiSetParamInt(handle, XI_PRM_FFS_FILE_ID, file_id);
xiGetParamInt(handle, XI_PRM_FFS_FILE_ID, &file_id);
```

Get list of files:

```
        int max_file_id = 0;
        xiGetParamInt(xiH, XI_PRM_FFS_FILE_ID XI_PRM_INFO_MAX, &max_file_id);
    char file_name[MAX_PATH];

    for(int i = 0; i <= max_file_id; i++)
    {
        memset(file_name, 0, MAX_PATH);
        stat = xiSetParamInt(xiH, XI_PRM_FFS_FILE_ID, i);
```

```
        stat = xiGetParamString(xiH, XI_PRM_FFS_FILE_NAME, file_name, MAX_PATH);
    }
```

XI_PRM_FFS_FILE_SIZE or "ffs_file_size"

Description: File size(file name can be set by XI_PRM_FFS_FILE_NAME).
Type: Integer. Read only.
Default: By default -.
Usage:

```
xiGetParamInt(handle, XI_PRM_FFS_FILE_SIZE, &file_size);
```

XI_PRM_FREE_FFS_SIZE or "free_ffs_size"

Description: Size of free camera FFS.
Type: Integer. Read only.
Default: By default -.
Usage:

```
xiGetParamInt(handle, XI_PRM_FREE_FFS_SIZE, &free_size);
```

XI_PRM_USED_FFS_SIZE or "used_ffs_size"

Description: Size of used camera FFS.
Type: Integer. Read only.
Default: By default -.
Usage:

```
xiGetParamInt(handle, XI_PRM_USED_FFS_SIZE, &used_size);
```

XI_PRM_FFS_ACCESS_KEY or "ffs_access_key"

Description: Setting of the key enables file operations on some cameras. It is required to set before usage of XI_PRM_WRITE_FILE_FFS.
Type: Integer.
Default: 0
Usage:

```
xiSetParamInt(handle, XI_PRM_FFS_ACCESS_KEY, 0x12345678);
```

Camera Lens Control

Some of XIMEA cameras can be equipped with controlled lens. API for lens control contains couple of parameters.
Lens tested OK with CB cameras (2017-04):

- CANON EF 50mm f/1.4 USM
- CANON EF 50mm f/1.8 II
- CANON EF 24-105 f4 L IS USM
- CANON EF 17-40mm f/4L USM
- CANON EF 100mm f/2.8 Macro USM
- CANON EF-S 17-55mm f/2.8 IS USM
- CANON EF 70-200mm f/4L IS USM
- CANON EF 50mm f/1.8 STM
- CANON EF 24mm f/2.8 STM
- CANON EF 10-18mm f/4.5-5.6 STM
- CANON EF 18-135mm f/3.5-5.6 STM
- Sigma 150mm f/2.8 EX DG OS HSM APO Macro
- Sigma 15mm f/2.8 EX DG

XI_PRM_LENS_MODE or lens_mode

Description: Status of lens control interface. This shall be set to XI_ON before any Lens operations.
Type: Integer.
Default: By default XI_OFF.
Usage:

```
xiSetParamInt(handle, XI_PRM_LENS_MODE, XI_ON);
```

XI_PRM_LENS_APERTURE_VALUE or lens_aperture_value

Description: Current lens aperture value in aperture stops. Examples: 2.8, 4, 5.6, 8, 11
Type: Float.

Default: 1
Usage:

```
xiSetParamFloat(handle, XI_PRM_LENS_APERTURE_VALUE, 5.6);
```

XI_PRM_LENS_FOCUS_MOVEMENT_VALUE or lens_focus_movement_value

Description: Lens current focus movement value to be used by XI_PRM_LENS_FOCUS_MOVE in motor steps. Positive numbers will direct the movement to infinity. Negative numbers will direct the movement to macro.
Type: Integer.
Default: 0
Usage:

```
xiSetParamInt(handle, XI_PRM_LENS_FOCUS_MOVEMENT_VALUE, 10);
```

XI_PRM_LENS_FOCUS_MOVE or lens_focus_move

Description: Moves lens focus motor by steps set in XI_PRM_LENS_FOCUS_MOVEMENT_VALUE.
Type: Command
Default: 0
Usage:

```
xiSetParamInt(handle, XI_PRM_LENS_MODE, XI_ON);
xiSetParamFloat(handle, XI_PRM_LENS_FOCUS_MOVEMENT_VALUE, 10);
xiSetParamInt(handle, XI_PRM_LENS_FOCUS_MOVE, 0); // move 10 steps to infinity
```

XI_PRM_LENS_FOCUS_DISTANCE or lens_focus_distance

Description: Lens focus distance in cm.
Type: Float
Default: 1000
Usage:

```
float distance_cm=0;
xiGetParamFloat(handle, XI_PRM_LENS_FOCUS_DISTANCE, &distance_cm);
```

XI_PRM_LENS_FOCAL_LENGTH or lens_focal_length

Description: Lens focal distance in mm. This parameter is constant for prime lens and can change in real time for zoom lens.
Type: Float
Default: 0
Usage:

```
float zoom_min_mm=0;
float zoom_max_mm=0;
xiGetParamFloat(handle, XI_PRM_LENS_FOCAL_LENGTH XI_PRM_INFO_MIN, &zoom_min_mm);
xiGetParamFloat(handle, XI_PRM_LENS_FOCAL_LENGTH XI_PRM_INFO_MAX, &zoom_max_mm);
```

XI_PRM_LENS_FEATURE_SELECTOR or lens_feature_selector

Description: Selects the current feature which is accessible by XI_PRM_LENS_FEATURE.
Type: Integer
Default: 0
Usage: See parameter XI_PRM_LENS_FEATURE
Possible values:

XI_PRM_LENS_FEATURE_SELECTOR	Value on XI_PRM_LENS_FEATURE
XI_LENS_FEATURE_MOTORIZED_FOCUS_SWITCH	Status of lens motorized focus switch
XI_LENS_FEATURE_MOTORIZED_FOCUS_BOUNDED	On read = 1 if motorized focus is on one of limits.
XI_LENS_FEATURE_MOTORIZED_FOCUS_CALIBRATION	On read = 1 if motorized focus is calibrated. Write 1 to start calibration.
XI_LENS_FEATURE_IMAGE_STABILIZATION_ENABLED	On read = 1 if image stabilization is enabled. Write 1 to enable image stabilization.
XI_LENS_FEATURE_IMAGE_STABILIZATION_SWITCH_STATUS	On read = 1 if image stabilization switch is in position On.
XI_LENS_FEATURE_IMAGE_ZOOM_SUPPORTED	On read = 1 if lens supports zoom = are not prime

XI_PRM_LENS_FEATURE or lens_feature

Description: Allows access to lens feature value currently selected by XI_PRM_LENS_FEATURE_SELECTOR.
Type: Integer
Default: 0
Usage:

```
xiSetParamInt(handle, XI_PRM_LENS_FEATURE_SELECTOR, XI_LENS_FEATURE_MOTORIZED_FOCUS_SWITCH);
int switch_status=0;
xiGetParamInt(handle, XI_PRM_LENS_FEATURE, &switch_status);
if (switch_status > 0)
   printf("The motorized focus switch on the lens is switched on.\n");
```

Sensor Feature Control

Some of XIMEA cameras has sensors with specific features

XI_PRM_SENSOR_FEATURE_SELECTOR or sensor_feature_selector

Description: Selects the current feature which is accessible by XI_PRM_SENSOR_FEATURE_VALUE.
Type: Integer.
Default: XI_SENSOR_FEATURE_ZEROROT_ENABLE
Usage:

```
xiSetParamInt(handle, XI_PRM_SENSOR_FEATURE_SELECTOR, XI_SENSOR_FEATURE_ZEROROT_ENABLE);
```

Selector	Camera model	Default value
XI_SENSOR_FEATURE_ZEROROT_ENABLE	MQ013xG-ON (PYTHON)	XI_OFF
XI_SENSOR_FEATURE_BLACK_LEVEL_CLAMP	MD	XI_OFF
XI_SENSOR_FEATURE_MD_FPGA_DIGITAL_GAIN_DISABLE	MD	XI_OFF
XI_SENSOR_FEATURE_ACQUISITION_RUNNING	CB,MC,MX,MT	XI_OFF

Selector description

- XI_SENSOR_FEATURE_ZEROROT_ENABLE
  When this feature is enabled the readout fof pixel line is overlapped with vertical shift of pixels from pixel are to readout shift registers. This effectively can eliminate time needed between consequent live readouts. The price for it is that the dynamic range is lower by 6dB compared to normal ROT.
- XI_SENSOR_FEATURE_ACQUISITION_RUNNING
  When this feature value is 1 - acquisition is running. Application can stop just acquisition of the sensor without freeing of image buffers by setting this value to 0. Then acquisition could be resumed by setting 1. It is also much faster compared to xiStopAcquisition/xiStartAcquisition.

This feature can be also added to CamTool.
Copy the file xiapi_camera_description_FULL.xml to *\XIMEA\XIMEACamTool64\camdesc* and replace the original file. When CamTool is opened, there are two xiAPI parameters of this feature (*Sensor feature selector*, *Sensor feature value*) in Performance tab on the right side (Visibility level: Guru). Select *Enable Zero ROT* and check *Sensor feature value*.

XI_PRM_SENSOR_FEATURE_VALUE or sensor_feature_value

Description: Allows access to sensor feature value currently selected by XI_PRM_SENSOR_FEATURE_SELECTOR.
Type: Integer.
Default: 0
Usage:

```
xiSetParamInt(handle, XI_PRM_SENSOR_FEATURE_SELECTOR, XI_SENSOR_FEATURE_ZEROROT_ENABLE);
xiSetParamInt(handle, XI_PRM_SENSOR_FEATURE_VALUE, XI_ON);
```

XI_PRM_ACQUISITION_STATUS_SELECTOR or “acquisition_status_selector”

Description: Selects the internal acquisition signal to read using XI_PRM_ACQUISITION_STATUS.
Type: Integer.
Default: All
Usage:

```
xiSetParamInt(handle, XI_PRM_ACQUISITION_STATUS_SELECTOR, XI_ACQUISITION_STATUS_ACQ_ACTIVE );
```

XI_PRM_ACQUISITION_STATUS or “acquisition_status”

Description: Returns status of acquisition.
Type: Integer.
Default: By default -.
Usage:

```
xiGetParam(handle, XI_PRM_ACQUISITION_STATUS, &value, sizeof(int), xiTypeInteger);
int xiGetParamInt(handle, XI_PRM_ACQUISITION_STATUS, &value);
```

API parameter modifiers

Description: The parameter modifiers allow you to acquire more information about the camera parameters (e.g. min. or max. value).
Also with certain parameters they allow direct update of these parameters without interrupting the image acquisition
loop (e.g. setting of exposure and gain).

XI_PRM_INFO_MIN

Description: Acquire parameter minimum value.
Usage:

```
xiGetParamInt(handle, XI_PRM_EXPOSURE XI_PRM_INFO_MIN, &exp_min);
```

XI_PRM_INFO_MAX

Description: Acquire parameter maximum value.
Usage:

```
xiGetParamInt(handle, XI_PRM_EXPOSURE XI_PRM_INFO_MAX, &exp_max);
```

XI_PRM_INFO_INCREMENT

Description: Get parameter possible increment step. The setting of value is limited to values MinumumValue+(N*IncrementValue)
Usage:

```
xiGetParamInt(handle, XI_PRM_HEIGHT XI_PRM_INFO_INCREMENT, &width_inc);
```

XI_PRM_INFO

Description: Acquire current parameter value. This modifier is **not mandatory**, **xiGetParam()** functions will return the current value even without this modifier.
Usage:

```
xiGetParamInt(handle, XI_PRM_EXPOSURE XI_PRM_INFO, &exp_val);
```

XI_PRMM_DIRECT_UPDATE

Description: Parameter modifier for direct update without stopping the streaming. Currently XI_PRM_EXPOSURE and XI_PRM_GAIN can be used with this modifier.
Usage:

```
xiSetParamInt(handle, XI_PRM_EXPOSURE XI_PRMM_DIRECT_UPDATE, exp_val);
xiSetParamInt(handle, XI_PRM_GAIN XI_PRMM_DIRECT_UPDATE, gain_val);
```

Image Buffers Queue

Functionality

The Image Buffers is first-in first-out (FIFO) type of queue.

Capturing

Each captured image is stored in the buffers queue. When application calls xiGetImage - the oldest image is removed from queue. Maximum number of images in queue can be set by parameter XI_PRM_BUFFERS_QUEUE_SIZE.

Flushing the queue

The images remain in the queue until they are overwritten or flushed. The queue is flushed on one of following conditions:

- acquisition is stopped (xiStopAcquisition)
- application set some of parameters using xiSetParam:
  - Exposure (XI_PRM_EXPOSURE) *Note
  - Gain (XI_PRM_GAIN) *Note
  - Downsampling (XI_PRM_DOWNSAMPLING)
  - Data Format (XI_PRM_IMAGE_DATA_FORMAT)
  - Width (XI_PRM_WIDTH)
  - Height (XI_PRM_HEIGHT)
  - Offset X (XI_PRM_OFFSET_X)
  - Offset Y (XI_PRM_OFFSET_Y)

*Note: Some of parameters can be changed without flushing the queue. It is possible to change parameter using modifier: XI_PRMM_DIRECT_UPDATE