Helpdesk XIMEA

xiAPI Manual

Support SK - 2024-04-15 - in xiAPI

https://www.ximea.com/support/wiki/apis/xiapi_manual

xiAPI Manual¶
Table of Contents¶

- XI_PRM_SENSOR_TAPS or "sensor_taps"
- XI_PRM_AEAG or "aeag"
- 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_SENS_DEFECTS_CORR_LIST_SELECTOR or "bpc_list_selector"
- XI_PRM_SENS_DEFECTS_CORR_LIST_CONTENT or "sens_defects_corr_list_content"
- XI_PRM_SENS_DEFECTS_CORR or "bpc"
- XI_PRM_AUTO_WB or "auto_wb"
- XI_PRM_MANUAL_WB or "manual_wb"
- XI_PRM_WB_ROI_OFFSET_X or "wb_roi_offset_x"
- XI_PRM_WB_ROI_OFFSET_Y or "wb_roi_offset_y"
- XI_PRM_WB_ROI_WIDTH or "wb_roi_width"
- XI_PRM_WB_ROI_HEIGHT or "wb_roi_height"
- XI_PRM_WB_KR or "wb_kr"
- XI_PRM_WB_KG or "wb_kg"
- XI_PRM_WB_KB or "wb_kb"
- XI_PRM_WIDTH or "width"
- XI_PRM_HEIGHT or "height"
- XI_PRM_OFFSET_X or "offsetX"
- XI_PRM_OFFSET_Y or "offsetY"
- XI_PRM_REGION_SELECTOR or "region_selector"
- XI_PRM_REGION_MODE or "region_mode"
- XI_PRM_HORIZONTAL_FLIP or "horizontal_flip"
- XI_PRM_VERTICAL_FLIP or "vertical_flip"
- XI_PRM_INTERLINE_EXPOSURE_MODE or "interline_exposure_mode"
- XI_PRM_FFC or "ffc"
- XI_PRM_FFC_FLAT_FIELD_FILE_NAME or "ffc_flat_field_file_name"
- XI_PRM_FFC_DARK_FIELD_FILE_NAME or "ffc_dark_field_file_name"
- XI_PRM_TOF_READOUT_MODE or "tof_readout_mode"
- XI_PRM_TOF_MODULATION_FREQUENCY or "tof_modulation_frequency"
- XI_PRM_TOF_MULTIPLE_PHASES_IN_BUFFER or "tof_multiple_phases_in_buffer"
- XI_PRM_TOF_PHASES_COUNT or "tof_phases_count"
- XI_PRM_TOF_PHASE_ANGLE or "tof_phase_angle"
- XI_PRM_TOF_PHASE_EXPOSURE_TIME or "tof_phase_exposure_time"
- XI_PRM_TOF_PHASE_SELECTOR or "tof_phase_selector"

Image Format

- XI_PRM_BINNING_SELECTOR or "binning_selector"

- XI_PRM_BINNING_VERTICAL_MODE or "binning_vertical_mode"
- XI_PRM_BINNING_VERTICAL or "binning_vertical"
- XI_PRM_BINNING_VERTICAL_FLOAT or "binning_vertical_float"
- XI_PRM_BINNING_HORIZONTAL_MODE or "binning_horizontal_mode"
- XI_PRM_BINNING_HORIZONTAL or "binning_horizontal"
- XI_PRM_BINNING_HORIZONTAL_FLOAT or "binning_horizontal_float"
- XI_PRM_BINNING_HORIZONTAL_PATTERN or "binning_horizontal_pattern"
- XI_PRM_BINNING_VERTICAL_PATTERN or "binning_vertical_pattern"
- XI_PRM_DECIMATION_SELECTOR or "decimation_selector"
- XI_PRM_DECIMATION_VERTICAL or "decimation_vertical"
- XI_PRM_DECIMATION_HORIZONTAL or "decimation_horizontal"
- XI_PRM_DECIMATION_HORIZONTAL_PATTERN or "decimation_horizontal_pattern"
- XI_PRM_DECIMATION_VERTICAL_PATTERN or "decimation_vertical_pattern"

AE Setup

- XI_PRM_EXP_PRIORITY or "exp_priority"
- XI_PRM_AG_MAX_LIMIT or "ag_max_limit"
- XI_PRM_AE_MAX_LIMIT or "ae_max_limit"
- XI_PRM_AEAG_LEVEL or "aeag_level"

Performance

- XI_PRM_LIMIT_BANDWIDTH or "limit_bandwidth"
- XI_PRM_LIMIT_BANDWIDTH_MODE or "limit_bandwidth_mode"
- XI_PRM_SENSOR_DATA_BIT_DEPTH or "sensor_bit_depth"
- XI_PRM_OUTPUT_DATA_BIT_DEPTH or "output_bit_depth"
- XI_PRM_IMAGE_DATA_BIT_DEPTH or "image_data_bit_depth"
- XI_PRM_OUTPUT_DATA_PACKING or "output_bit_packing"
- XI_PRM_OUTPUT_DATA_PACKING_TYPE or "output_bit_packing_type"

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_TEMP_CONTROL_MODE or "device_temperature_ctrl_mode"
- 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"
- XI_PRM_SENSOR_BOARD_TEMP or "sensor_board_temp"
- XI_PRM_TEMP_ELEMENT_SEL or "device_temperature_element_sel"

- XI_PRM_TEMP_ELEMENT_VALUE or "device_temperature_element_val"

Color Correction

- 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"
- XI_PRM_IMAGE_IS_COLOR or "iscolor"
- XI_PRM_COLOR_FILTER_ARRAY or "cfa"
- XI_PRM_GAMMAY or "gammaY"
- XI_PRM_GAMMAC or "gammaC"
- XI_PRM_SHARPNESS or "sharpness"
- XI_PRM_CC_MATRIX_00 or "ccMTX00"
- XI_PRM_DEFAULT_CC_MATRIX or "defccMTX"
- XI_PRM_CC_MATRIX_NORM or "ccMTXnorm"

Device IO

- XI_PRM_TRG_SOURCE or "trigger_source"
- XI_PRM_TRG_SOFTWARE or "trigger_software"
- XI_PRM_TRG_SELECTOR or "trigger_selector"
- XI_PRM_TRG_OVERLAP or "trigger_overlap"
- XI_PRM_ACQ_FRAME_BURST_COUNT or "acq_frame_burst_count"
- XI_PRM_TIMESTAMP or "timestamp"

GPIO Setup

- XI_PRM_GPI_SELECTOR or "gpi_selector"
- XI_PRM_GPI_MODE or "gpi_mode"
- XI_PRM_GPI_LEVEL or "gpi_level"
- XI_PRM_GPI_LEVEL_AT_IMAGE_EXP_START or "gpi_level_at_image_exp_start"
- XI_PRM_GPI_LEVEL_AT_IMAGE_EXP_END or "gpi_level_at_image_exp_end"
- 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_DEBOUNCE_EN or "dbnc_en"

Debounce Setup

- XI_PRM_DEBOUNCE_T0 or "dbnc_t0"
- XI_PRM_DEBOUNCE_T1 or "dbnc_t1"
- XI_PRM_DEBOUNCE_POL or "dbnc_pol"

Lens Control

- XI_PRM_LENS_MODE or "lens_mode"
- XI_PRM_LENS_APERTURE_VALUE or "lens_aperture_value"
- XI_PRM_LENS_APERTURE_INDEX or "lens_aperture_index"
- XI_PRM_LENS_FOCUS_MOVEMENT_VALUE or "lens_focus_movement_value"
- XI_PRM_LENS_FOCUS_MOVE or "lens_focus_move"
- XI_PRM_LENS_FOCAL_LENGTH or "lens_focal_length"
- XI_PRM_LENS_FEATURE_SELECTOR or "lens_feature_selector"
- XI_PRM_LENS_FEATURE or "lens_feature"

Device info parameters

- XI_PRM_DEVICE_NAME or "device_name"
- XI_PRM_DEVICE_TYPE or "device_type"
- XI_PRM_DEVICE_MODEL_ID or "device_model_id"
- XI_PRM_SENSOR_MODEL_ID or "sensor_model_id"
- XI_PRM_DEVICE_SN or "device_sn"
- XI_PRM_DEVICE_SENS_SN or "device_sens_sn"
- XI_PRM_DEVICE_INSTANCE_PATH or "device_inst_path"
- XI_PRM_DEVICE_LOCATION_PATH or "device_loc_path"
- XI_PRM_DEVICE_USER_ID or "device_user_id"
- XI_PRM_DEVICE_MANIFEST or "device_manifest"
- XI_PRM_IMAGE_USER_DATA or "image_user_data"

Device acquisition settings

- XI_PRM_IMAGE_DATA_FORMAT_RGB32_ALPHA or "imgdataformatrgb32alpha"
- XI_PRM_IMAGE_PAYLOAD_SIZE or "imgpayloadsize"
- XI_PRM_TRANSPORT_PIXEL_FORMAT or "transport_pixel_format"
- XI_PRM_TRANSPORT_DATA_TARGET or "transport_data_target"
- XI_PRM_SENSOR_CLOCK_FREQ_HZ or "sensor_clock_freq_hz"
- XI_PRM_SENSOR_CLOCK_FREQ_INDEX or "sensor_clock_freq_index"
- XI_PRM_SENSOR_OUTPUT_CHANNEL_COUNT or "sensor_output_channel_count"
- XI_PRM_FRAMERATE or "framerate"
- XI_PRM_COUNTER_SELECTOR or "counter_selector"
- XI_PRM_COUNTER_VALUE or "counter_value"
- XI_PRM_ACQ_TIMING_MODE or "acq_timing_mode"
- XI_PRM_AVAILABLE_BANDWIDTH or "available_bandwidth"
- XI_PRM_BUFFER_POLICY or "buffer_policy"
- XI_PRM_LUT_EN or "LUTEnable"
- XI_PRM_LUT_INDEX or "LUTIndex"
- XI_PRM_LUT_VALUE or "LUTValue"
- XI_PRM_TRG_DELAY or "trigger_delay"

- XI_PRM_TS_RST_MODE or "ts_rst_mode"
- XI_PRM_TS_RST_SOURCE or "ts_rst_source"

Extended Device parameters

- XI_PRM_IS_DEVICE_EXIST or "isexist"
- 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_DEVICE_RESET or "device_reset"
- XI_PRM_CONCAT_IMG_MODE or "concat_img_mode"
- XI_PRM_CONCAT_IMG_COUNT or "concat_img_count"
- XI_PRM_CONCAT_IMG_TRANSPORT_IMG_OFFSET or "concat_img_transport_img_offset"
- XI_PRM_PROBE_SELECTOR or "probe_selector"
- XI_PRM_PROBE_VALUE or "probe_value"

Sensor Defects Correction

- XI_PRM_COLUMN_FPN_CORRECTION or "column_fpn_correction"
- XI_PRM_ROW_FPN_CORRECTION or "row_fpn_correction"
- XI_PRM_COLUMN_BLACK_OFFSET_CORRECTION or "column_black_offset_correction"
- XI_PRM_ROW_BLACK_OFFSET_CORRECTION or "row_black_offset_correction"

Sensor features

- XI_PRM_SENSOR_MODE or "sensor_mode"
- XI_PRM_HDR or "hdr"
- XI_PRM_HDR_KNEEPOINT_COUNT or "hdr_kneepoint_count"
- XI_PRM_HDR_T1 or "hdr_t1"
- XI_PRM_HDR_T2 or "hdr_t2"
- XI_PRM_KNEEPOINT1 or "hdr_kneepoint1"
- XI_PRM_KNEEPOINT2 or "hdr_kneepoint2"
- XI_PRM_IMAGE_BLACK_LEVEL or "image_black_level"
- XI_PRM_IMAGE_AREA or "image_area"
- XI_PRM_DUAL_ADC_MODE or "dual_adc_mode"
- XI_PRM_DUAL_ADC_GAIN_RATIO or "dual_adc_gain_ratio"
- XI_PRM_DUAL_ADC_THRESHOLD or "dual_adc_threshold"
- XI_PRM_COMPRESSION_REGION_SELECTOR or "compression_region_selector"
- XI_PRM_COMPRESSION_REGION_START or "compression_region_start"

- XI_PRM_COMPRESSION_REGION_GAIN or "compression_region_gain"

Version info

- XI_PRM_VERSION_SELECTOR or "version_selector"
- XI_PRM_VERSION or "version"
- XI_PRM_API_VERSION or "api_version"
- XI_PRM_DRV_VERSION or "drv_version"
- XI_PRM_MCU1_VERSION or "version_mcu1"
- XI_PRM_MCU2_VERSION or "version_mcu2"
- XI_PRM_MCU3_VERSION or "version_mcu3"
- XI_PRM_FPGA1_VERSION or "version_fpga1"
- XI_PRM_XMLMAN_VERSION or "version_xmlman"
- XI_PRM_HW_REVISION or "hw_revision"
- XI_PRM_FACTORY_SET_VERSION or "factory_set_version"

API features

- XI_PRM_DEBUG_LEVEL or "debug_level"
- XI_PRM_AUTO_BANDWIDTH_CALCULATION or "auto_bandwidth_calculation"
- XI_PRM_NEW_PROCESS_CHAIN_ENABLE or "new_process_chain_enable"
- XI_PRM_PROC_NUM_THREADS or "proc_num_threads"

Camera FFS

- XI_PRM_READ_FILE_FFS or "read_file_ffs"
- XI_PRM_WRITE_FILE_FFS or "write_file_ffs"
- XI_PRM_FFS_FILE_NAME or "ffs_file_name"
- XI_PRM_FFS_FILE_ID or "ffs_file_id"
- XI_PRM_FFS_FILE_SIZE or "ffs_file_size"
- XI_PRM_FREE_FFS_SIZE or "free_ffs_size"
- XI_PRM_USED_FFS_SIZE or "used_ffs_size"
- XI_PRM_FFS_ACCESS_KEY or "ffs_access_key"

APIContextControl

- XI_PRM_API_CONTEXT_LIST or "xiapi_context_list"

Sensor Control

- XI_PRM_SENSOR_FEATURE_SELECTOR or "sensor_feature_selector"
- XI_PRM_SENSOR_FEATURE_VALUE or "sensor_feature_value"

Extended Features

- XI_PRM_ACQUISITION_STATUS_SELECTOR or "acquisition_status_selector"
- XI_PRM_ACQUISITION_STATUS or "acquisition_status"
- XI_PRM_DP_UNIT_SELECTOR or "dp_unit_selector"

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);
```

Note: First call of this function enumerates all connected cameras. When the number of cameras is changing during the execution of a program, we recommend you to call *xiGetNumberDevices* function each time you call additional *xiOpenDevice*.

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);
```

Corresponding Enumerator XI_OPEN_BY

type	representing value	result
XI_OPEN_BY_INST_PATH	0	Open camera by its hardware path
XI_OPEN_BY_SN	1	Open camera by its serial number
XI_OPEN_BY_USER_ID	2	open camera by its custom user ID
XI_OPEN_BY_LOC_PATH	3	Open camera by its hardware location path

Note1: When the value of the parameter XI_PRM_DEVICE_USER_IDis changed and we want to open the camera with function xiOpenDeviceBy with the value XI_PRM_DEVICE_USER_ID, it is necessary to do a power cycle on this camera beforehand. This affects only cameras with USB data interface.

Note2: First call of this function enumerates all connected cameras. When the number of cameras is changing during the execution of a program, we recommend you to call xiGetNumberDevices function each time you call additional xiOpenDeviceBy.

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. 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 data. (see Note1)
4	bp_size	Filled buffer size. (see Note2)
4	frm	Format of image data get from GetImage.
4	width	width of incoming image.
4	height	height of incoming image.
4	nframe	Frame number. On some cameras it is reset by exposure, gain, downsampling change, auto exposure (AEAG).
4	tsSec	Seconds part of image timestamp (see Note3).
4	tsUSec	Micro-seconds part image timestamp (see Note3). Range 0-999999 us.
4	GPI_level	Levels of digital inputs/outputs of the camera at time of exposure start/end (sample time and bits are specific for each camera model)
4	black_level	Black level of image (ONLY for MONO and RAW formats). (see Note4)
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	transport_frm	Current format of pixels on transport layer.
x	img_desc	description of image areas and format.
4	DownsamplingX	Horizontal downsampling
4	DownsamplingY	Vertical downsampling
4	flags	description of XI_IMG.
4	exposure_time_us	Exposure time of this image in microseconds. (see Note5)
4	gain_db	Gain used for this image in deci-bells. (see Note6)
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	(see Note7)
20	exposure_sub_times_us	(see Note8)
	data_saturation	Pixel value of saturation
4	wb_red	Red coefficient of white balance
4	wb_green	Green coefficient of white balance
4	wb_blue	Blue coefficient of white balance
4	lg_black_level	In case of multi gain channel readout, the black level low gain channel
4	hg_black_level	In case of multi gain channel readout, the black level high gain channel
4	lg_range	In case of multi gain channel readout, the valid range of low gain channel
4	hg_range	In case of multi gain channel readout, the valid range of high gain channel
4	gain_ratio	Gain ratio for dual-channel modes (high_gain_channel/low_gain_channel). Unitless.

4	fDownsamplingX	Horizontal downsampling
4	fDownsamplingY	Vertical downsampling
	color_filter_array	Mosaic of tiny color filters placed over the pixel sensors of an image sensor.
4	tof_phases_count	Number of phases of ToF sensor. E.g. 4 for four phases.
4	tof_phase_id	Current phase ID of ToF sensor data starting from 1
4	tof_multiple_phases_in_buffer	Is multiple phases in buffer (bp)
	data_sign_mode	Sign mode or signedness is a property of data.

Note1: If the buffer policy is set to XI_BP_UNSAFE, the bp is set to the buffer allocated by API. If set to XI_BP_SAFE, the data is copied to bp, which should be allocated by the application.

Note2: If the buffer policy is set to XI_BP_SAFE, xiGetImage fills this field with the current size of the received image data.

Note3: Depending on the camera family, the TimeStamp is represented as a counter:

- xiQ, xiD: 40-bit microsecond number - (overlaps after 305 hours)
- xiC, xiB, xiT, xiX: 64-bit 4 nanosecond number (overlaps after 2339 years)

This counter is converted to image header fields **tsSec** and **tsUSec.**

TimeStamp on **xiQ, xiD** is recorded at the start of Data Readout.

TimeStamp on **xiC, xiB, xiX, xiT** is recorded at the start of Exposure.

TimeStamp is **NOT** implemented on some cameras (e.g. xiMU - MU9), in which case the image header contains only a constant number instead of a valid TimeStamp.

Note4: xiQ cameras report calculated black_level. We do not guarantee the accuracy of black_level calculation when exposure time exceeds 50ms and/or gain is above 3dB.

Note5: Some camera models (MQ, MU) might report this exposure time earlier before exposure is applied to image. Cameras with IMX sensors (MC, MX, MT) report value measured by the FPGA - this value is systemically lower than the value returned by xiGetParam with XI_PRM_EXPOSURE parameter; difference is typically below 20 us.

Note6: Valid only for MQ, MD, and MR camera families with the following conditions:

- If a gain parameter is changed while the sensor is idle (not exposing nor reading out) the gain is valid for the next image.
- If a gain parameter is changed while the sensor is busy (exposing or reading out) and

the 'direct_update' modifier is not used, the gain is valid for the next image.
- If a gain parameter is changed while the sensor is busy (exposing or reading out) and the 'direct_update' modifier is used, the gain value in the header might be incorrect after the change for the next 1-2 images. This is caused by the asynchronous setting of sensor registers and the frame acquisition process.

Note7: Available only on PCIe cameras (CB,MX). ImageUserData is controlled by a user application using ImageUserData or XI_PRM_IMAGE_USER_DATA parameter.

Note8: Array with five substitute exposure times in microseconds used by XI_TRG_SEL_MULTIPLE_EXPOSURES or hardware controlled HDR.

GPI_level-Sampling-in-Header¶

Digital Inputs are sampled on some cameras.

- MQ: at exposure end
- CB,MX,MC: at exposure start and exposure end
  The samples are available after calling of xiGetImage using the parameters XI_PRM_GPI_LEVEL_AT_IMAGE_EXP_START, XI_PRM_GPI_LEVEL_AT_IMAGE_EXP_END. See example at the parameters description.

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:** Default value, however it can differ between camera models.
- **Is invalidated by:** List of parameters. Changing any of the listed parameters may lead to the update of value or range (min, max, increment) of the respective parameter.
- **Usage:** Example of usage of this parameter in application (C code)

Basic¶

XI_PRM_EXPOSURE or "exposure"¶

Description: Current exposure time in microseconds. When parameter is set by xiSetParam the API checks the range. If it is within the range, it tries to find the closest settable value and set it. The actual value can be read by xiGetParam. E.g. Application set exposure time to 1000us, however closest possible value is 1029us (it is typically based on sensor line read-out period). This value is accessible over xiGetParam.

identifiers: SENSOR

Type: Float.
Default value: 1000.0

Usage:

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

XI_PRM_EXPOSURE_TIME_SELECTOR or "exposure_time_selector"¶

Description: Selector for Exposure parameter

Type: Enumerator.
Default value: XI_EXPOSURE_TIME_SELECTOR_COMMON

Usage:

```
int exposure_time_selector = 0;
xiGetParamInt(handle, XI_PRM_EXPOSURE_TIME_SELECTOR,
&exposure_time_selector);
xiSetParamInt(handle, XI_PRM_EXPOSURE_TIME_SELECTOR,
XI_EXPOSURE_TIME_SELECTOR_COMMON);
```

Value	Description
XI_EXPOSURE_TIME_SELECTOR_COMMON	Selects the common Exposure Time
XI_EXPOSURE_TIME_SELECTOR_GROUP1	Selects the common Exposure Time for pixel group 1 (for InterlineExposureMode)
XI_EXPOSURE_TIME_SELECTOR_GROUP2	Selects the common Exposure Time for pixel group 2 (for InterlineExposureMode)
XI_EXPOSURE_TIME_SELECTOR_DUAL_TRG_EXP_ZONE_1	Selects the Exposure Time for Zone 1 (for Dual Trigger Exposure feature)

XI_EXPOSURE_TIME_SELECTOR_DUAL_TRG_EXP_ZONE_2	Selects the Exposure Time for Zone 2 (for Dual Trigger Exposure feature)

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. See more details in article Multiple exposures in one frame.

Note: This setting is valid only if the trigger selector is set to ExposureActive or ExposureStart.

Supported cameras: MC031xG-SY, MC050xG-SY, MC089xG-SY, MC124xG-SY, MX031xG-SY, MX050xG-SY, MX089xG-SY, MX124xG-SY, MT031xG-SY, MT050xG-SY

Type: Integer.
Default value: 1

Usage:

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

Selector XI_GAIN_SELECTOR_ALL is mapped on most cameras to analog gain.

Type: Enumerator.
Default value: XI_GAIN_SELECTOR_ANALOG_ALL

Usage:

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

Value	Description
XI_GAIN_SELECTOR_ALL	Gain selector selects all channels. Implementation of gain type depends on camera.
XI_GAIN_SELECTOR_ANALOG_ALL	Gain selector selects all analog channels. This is available only on some cameras.
XI_GAIN_SELECTOR_DIGITAL_ALL	Gain selector selects all digital channels. This is available only on some cameras.

XI_GAIN_SELECTOR_ANALOG_TAP1	Gain selector selects tap 1. This is available only on some cameras.
XI_GAIN_SELECTOR_ANALOG_TAP2	Gain selector selects tap 2. This is available only on some cameras.
XI_GAIN_SELECTOR_ANALOG_TAP3	Gain selector selects tap 3. This is available only on some cameras.
XI_GAIN_SELECTOR_ANALOG_TAP4	Gain selector selects tap 4. This is available only on some cameras.
XI_GAIN_SELECTOR_ANALOG_N	First of two channels of programmable gain control (PGC) function - Gain setting of R, B pixels (North column analog gain). This is available only on some cameras.
XI_GAIN_SELECTOR_ANALOG_S	Second of two channels of programmable gain control (PGC) function - Gain setting of Gr, Gb pixels (South column analog gain). This is available only on some cameras.

XI_PRM_GAIN or "gain"¶

Description: Current gain in dB. When parameter is set by xiSetParam the API checks the range. If it is within the range, it tries to find the closest settable value and set it. The actual value can be read by xiGetParam. E.g. Application set gain to 1.3dB, however closest possible value is 1.35dB (analog gain is typically based on sensor PGA registers capabilities). This value is accessible over xiGetParam.

Type: Float.
Default value: 0.0

Is invalidated by: XI_PRM_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING, XI_PRM_BINNING_VERTICAL, XI_PRM_BINNING_HORIZONTAL, XI_PRM_DECIMATION_VERTICAL, XI_PRM_DECIMATION_HORIZONTAL, XI_PRM_DP_PARAM_VALUE, XI_PRM_HDR, XI_PRM_SENSOR_DATA_BIT_DEPTH, XI_PRM_USER_SET_LOAD
Usage:

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

XI_PRM_DOWNSAMPLING or "downsampling"¶

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

Note1: Downsampling can be changed only before an acquisition is started.

Note2: Changing this parameter will flush all images from the buffer queue.

Type: Enumerator.

Default value: 1

Usage:

```
xiSetParamInt(handle, XI_PRM_DOWNSAMPLING, XI_DWN_2x2);
```

Value	Description
XI_DWN_1x1	1 sensor pixel = 1 image pixel
XI_DWN_2x2	2x2 sensor pixels = 1 image pixel
XI_DWN_3x3	Downsampling 3x3.
XI_DWN_4x4	4x4 sensor pixels = 1 image pixel
XI_DWN_5x5	Downsampling 5x5.
XI_DWN_6x6	Downsampling 6x6.
XI_DWN_7x7	Downsampling 7x7.
XI_DWN_8x8	Downsampling 8x8.
XI_DWN_9x9	Downsampling 9x9.
XI_DWN_10x10	Downsampling 10x10.
XI_DWN_16x16	Downsampling 16x16.

XI_PRM_DOWNSAMPLING_TYPE or "downsampling_type"¶
Description: Changes image downsampling type (binning or skipping).

Note1: Changing this parameter will remove all images from the buffer queue.

Note2: Changing this parameter will remove all images from the buffer queue.

Type: Enumerator.
Default value: XI_BINNING

Usage:

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

Value	Description
XI_BINNING	pixels are interpolated - better image
XI_SKIPPING	pixels are skipped - higher frame rate

XI_PRM_TEST_PATTERN_GENERATOR_SELECTOR or "test_pattern_generator_selector"¶

Description: Selects Test Pattern Generator Engine.

Type: Enumerator.
Default value: XI_TESTPAT_GEN_SENSOR

Usage:

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

Value	Description
XI_TESTPAT_GEN_SENSOR	Sensor test pattern generator
XI_TESTPAT_GEN_FPGA	FPGA Test Pattern Generator
XI_TESTPAT_GEN_MCU	MCU Test Pattern Generator

XI_PRM_TEST_PATTERN or "test_pattern"¶

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

Type: Enumerator.
Default value: XI_TESTPAT_OFF

Is invalidated by: XI_PRM_TEST_PATTERN_GENERATOR_SELECTOR
Usage:

```
int test_pattern = 0;
xiGetParamInt(handle, XI_PRM_TEST_PATTERN, &test_pattern);
xiSetParamInt(handle, XI_PRM_TEST_PATTERN, XI_TESTPAT_OFF);
```

Value	Description
XI_TESTPAT_OFF	Testpattern turned off.
XI_TESTPAT_BLACK	Image is filled with darkest possible image.
XI_TESTPAT_WHITE	Image is filled with brightest possible image.
XI_TESTPAT_GREY_HORIZ_RAMP	Image is filled horizontally with an image that goes from the darkest possible value to the brightest.
XI_TESTPAT_GREY_VERT_RAMP	Image is filled vertically with an image that goes from the darkest possible value to the brightest.

XI_TESTPAT_GREY_HORIZ_RAMP_MOVING	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	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	A moving horizontal line is superimposed on the live image.
XI_TESTPAT_VERT_LINE_MOVING	A moving vertical line is superimposed on the live image.
XI_TESTPAT_COLOR_BAR	Image is filled with stripes of color including White, Black, Red, Green, Blue, Cyan, Magenta and Yellow.
XI_TESTPAT_FRAME_COUNTER	A frame counter is superimposed on the live image.
XI_TESTPAT_DEVICE_SPEC_COUNTER	128bit counter.

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

In enumerators table second value in comment bar stands for one pixel data in memory [one_byte].

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

Format: XI_MONO8 **Parameters controlled automatically:**

- XI_PRM_SENSOR_DATA_BIT_DEPTH = 8 (see Note1)
- XI_PRM_OUTPUT_DATA_BIT_DEPTH = 8 (see Note1)
- XI_PRM_OUTPUT_DATA_PACKING = OFF

Image Processing: enabled

Format: XI_RAW8 **Parameters controlled automatically:**

- XI_PRM_SENSOR_DATA_BIT_DEPTH = 8 (see Note1)
- XI_PRM_OUTPUT_DATA_BIT_DEPTH = 8 (see Note1)
- XI_PRM_OUTPUT_DATA_PACKING = OFF

Image Processing: disabled

Format: XI_MONO16 (see Note2) **Parameters controlled automatically:**

- XI_PRM_SENSOR_DATA_BIT_DEPTH = maximum
- XI_PRM_OUTPUT_DATA_BIT_DEPTH = SENSOR_DATA_BIT_DEPTH
- XI_PRM_OUTPUT_DATA_PACKING = ON (see Note1)

Image Processing: enabled

Format: XI_RAW16 (see Note2) **Parameters controlled automatically:**

- XI_PRM_SENSOR_DATA_BIT_DEPTH = maximum
- XI_PRM_OUTPUT_DATA_BIT_DEPTH = SENSOR_DATA_BIT_DEPTH
- XI_PRM_OUTPUT_DATA_PACKING = ON (see Note1)

Image Processing: disabled
Format: XI_RGB32, XI_RGB24, XI_RGB_PLANAR **Parameters controlled automatically:**

- XI_PRM_SENSOR_DATA_BIT_DEPTH = maximum
- XI_PRM_OUTPUT_DATA_BIT_DEPTH = SENSOR_DATA_BIT_DEPTH
- XI_PRM_OUTPUT_DATA_PACKING = ON (see Note1)

Image Processing: enabled
Note1: Only if camera implementation allows this mode.

Note2: 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 () might change. Please check or set image resolution after changing of Image Data Format.

Note3: Bits alignment: Values are aligned to LSB.

- sensor bits per pixel: **10** >>> values in mode XI_RAW16: **0-1023**
- sensor bits per pixel: **12** >>> values in mode XI_RAW16: **0-4095**
- sensor bits per pixel: **14** >>> values in mode XI_RAW16: **0-16383**

Example: Camera produces 10 bits data and data format XI_RAW16bit is selected - each 16bit word (pixel) can contain values in range 0-1023.

Note4: 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.

Type: Enumerator.
Default value: XI_MONO8

Usage:

```
int imgdataformat = 0;
xiGetParamInt(handle, XI_PRM_IMAGE_DATA_FORMAT, &imgdataformat);
xiSetParamInt(handle, XI_PRM_IMAGE_DATA_FORMAT, XI_MONO8);
```

Value	Description
XI_MONO8	8 bits per pixel. [Intensity] (see Note5,Note6)
XI_MONO16	16 bits per pixel. [Intensity LSB] [Intensity MSB] (see Note5,Note6)
XI_RGB24	RGB data format. [Blue][Green][Red] (see Note5)
XI_RGB32	RGBA data format. [Blue][Green][Red][0] (see Note5)
XI_RGB_PLANAR	RGB planar data format. [Red][Red]...[Green][Green]...[Blue][Blue]... (see Note5)
XI_RAW8	8 bits per pixel raw data from sensor. [pixel byte] raw data from transport (camera output)
XI_RAW16	16 bits per pixel raw data from sensor. [pixel byte low] [pixel byte high] 16 bits (depacked) raw data
XI_FRM_TRANSPORT_DATA	Data from transport layer (e.g. packed). Depends on data on the transport layer (see Note7)
XI_RGB48	RGB data format. [Blue low byte][Blue high byte][Green low][Green high][Red low][Red high] (see Note5)
XI_RGB64	RGBA data format. [Blue low byte][Blue high byte][Green low][Green high][Red low][Red high][0][0] (Note5)
XI_RGB16_PLANAR	RGB16 planar data format
XI_RAW8X2	8 bits per pixel raw data from sensor(2 components in a row). [ch1 pixel byte] [ch2 pixel byte] 8 bits raw data from 2 channels (e.g. high gain and low gain channels of sCMOS cameras)
XI_RAW8X4	8 bits per pixel raw data from sensor(4 components in a row). [ch1 pixel byte [ch2 pixel byte] [ch3 pixel byte] [ch4 pixel byte] 8 bits raw data from 4 channels (e.g. sCMOS cameras)
XI_RAW16X2	16 bits per pixel raw data from sensor(2 components in a row). [ch1 pixel byte low] [ch1 pixel byte high] [ch2 pixel byte low] [ch2 pixel byte high] 16 bits (depacked) raw data from 2 channels (e.g. high gain and low gain channels of sCMOS cameras)

XI_RAW16X4	16 bits per pixel raw data from sensor(4 components in a row). [ch1 pixel byte low] [ch1 pixel byte high] [ch2 pixel byte low] [ch2 pixel byte high] [ch3 pixel byte low] [ch3 pixel byte high] [ch4 pixel byte low] [ch4 pixel byte high] 16 bits (depacked) raw data from 4 channels (e.g. sCMOS cameras)
XI_RAW32	32 bits per pixel raw data from sensor in integer format (LSB first). 4 bytes (LSB first) pixel (depacked) raw data
XI_RAW32FLOAT	32 bits per pixel raw data from sensor in single-precision floating point format. 4 bytes per pixel (depacked) raw data

Note5: 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.

Note6: 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.

Note7: When using Transport Data Format, the Image Processing block from XiAPI 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_SIGN or "image_data_sign"¶
Description: Signedness of image data.

Type: Enumerator.
Default value: XI_DATA_SM_UNSIGNED

Is invalidated by: XI_PRM_IMAGE_DATA_FORMAT, XI_PRM_TOF_READOUT_MODE
Usage:

```
int image_data_sign = 0;
xiGetParamInt(handle, XI_PRM_IMAGE_DATA_SIGN,
&image_data_signsizeof(value));
```

Value	Description
XI_DATA_SM_UNSIGNED	Unsigned if it can only represent non-negative numbers (zero or positive numbers).
XI_DATA_SM_SIGNED_2C	Signed if it can represent both positive and negative numbers (two's complement).
XI_DATA_SM_SIGNED_FLOATING	Signed floating point data type.

XI_PRM_SHUTTER_TYPE or "shutter_type"¶
Description: Type of sensor shutter.

Type: Enumerator.
Default value: XI_SHUTTER_GLOBAL

Is invalidated by: XI_PRM_TRG_SOURCE
Usage:

```
int shutter_type = 0;
xiGetParamInt(handle, XI_PRM_SHUTTER_TYPE, &shutter_type);
xiSetParamInt(handle, XI_PRM_SHUTTER_TYPE, XI_SHUTTER_GLOBAL);
```

Value	Description
XI_SHUTTER_GLOBAL	Sensor Global Shutter(CMOS sensor)
XI_SHUTTER_ROLLING	Sensor Electronic Rolling Shutter(CMOS sensor)
XI_SHUTTER_GLOBAL_RESET_RELEASE	Sensor Global Reset Release Shutter(CMOS sensor)

XI_PRM_SENSOR_TAPS or "sensor_taps"¶
Description: Set/get the number of taps used on sensor.

Type: Enumerator.
Default value: 1

Usage:

```
int sensor_taps = 0;
xiGetParamInt(handle, XI_PRM_SENSOR_TAPS, &sensor_taps);
xiSetParamInt(handle, XI_PRM_SENSOR_TAPS, XI_TAP_CNT_1);
```

Value	Description
XI_TAP_CNT_1	1 sensor tap selected.
XI_TAP_CNT_2	2 sensor taps selected.
XI_TAP_CNT_4	4 sensor taps selected.

XI_PRM_AEAG or "aeag"¶
Description: Automatic exposure/gain.

Type: Integer.
Default value: XI_OFF

Usage:

```
int value = 0;
```

```
xiGetParamInt(handle, XI_PRM_AEAG, &value);
xiSetParamInt(handle, XI_PRM_AEAG, XI_ON);
```

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: Integer.
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_AEAG_ROI_OFFSET_X, &value);
xiSetParamInt(handle, XI_PRM_AEAG_ROI_OFFSET_X, value);
```

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: Integer.
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_AEAG_ROI_OFFSET_Y, &value);
xiSetParamInt(handle, XI_PRM_AEAG_ROI_OFFSET_Y, value);
```

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: Integer.
Default value: Depends on the sensors resolution and downsampling.

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_AEAG_ROI_WIDTH, &value);
xiSetParamInt(handle, XI_PRM_AEAG_ROI_WIDTH, value);
```

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: Integer.
Default value: Depends on the sensors resolution and downsampling.

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_AEAG_ROI_HEIGHT, &value);
xiSetParamInt(handle, XI_PRM_AEAG_ROI_HEIGHT, value);
```

XI_PRM_SENS_DEFECTS_CORR_LIST_SELECTOR or "bpc_list_selector"¶
Description: Selector for current sensor defects list used by Sensor Defect Correction. For more information see Sensor Defect Correction support page.

Type: Enumerator.
Default value: XI_SENS_DEFFECTS_CORR_LIST_SEL_FACTORY

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);
```

Value	Description
XI_SENS_DEFFECTS_CORR_LIST_SEL_FACTORY	Factory defect correction list
XI_SENS_DEFFECTS_CORR_LIST_SEL_USER0	User defect correction list
XI_SENS_DEFFECTS_CORR_LIST_SEL_IN_CAMERA	Device specific defect correction list

XI_PRM_SENS_DEFECTS_CORR_LIST_CONTENT or "sens_defects_corr_list_content"¶
Description: Set/Get current sensor defects list used by Sensor Defect Correction(in specific text format).

Type: String.
Default value: -

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. For more information see Sensor Defect Correction support page.

Type: Integer.
Default value: XI_OFF

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_SENS_DEFECTS_CORR, &value);
xiSetParamInt(handle, XI_PRM_SENS_DEFECTS_CORR, XI_ON);
```

XI_PRM_AUTO_WB or "auto_wb"¶

Description: Automatic white balance.

Type: Integer.
Default value: XI_OFF

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_AUTO_WB, &value);
xiSetParamInt(handle, XI_PRM_AUTO_WB, XI_ON);
```

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 value: 0

Usage:

```
// now camera should see the white color in approximately 50% of
level
xiSetParamInt(handle, XI_PRM_MANUAL_WB, 1);
```

```
xiGetImage(handle, 1000, &image); // now API automatically calculates
the white balance
xiGetImage(handle, 1000, &image); // this and next images will have
corrected white balance
```

XI_PRM_WB_ROI_OFFSET_X or "wb_roi_offset_x"¶

Description: X offset of the area used for manual WB calculation. The sum of XI_PRM_WB_ROI_OFFSET_X and XI_PRM_WB_ROI_WIDTH must be equal or lower than the image resolution(width).

Supported cameras: MC,CB,MX

Type: Integer.
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_WB_ROI_OFFSET_X, &value);
xiSetParamInt(handle, XI_PRM_WB_ROI_OFFSET_X, value);
```

XI_PRM_WB_ROI_OFFSET_Y or "wb_roi_offset_y"¶

Description: Y offset of the area used for manual WB calculation. The sum of XI_PRM_WB_ROI_OFFSET_Y and XI_PRM_WB_ROI_HEIGHT must be equal or lower than the image resolution(height).

Supported cameras: MC,CB,MX

Type: Integer.
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_WB_ROI_OFFSET_Y, &value);
xiSetParamInt(handle, XI_PRM_WB_ROI_OFFSET_Y, value);
```

XI_PRM_WB_ROI_WIDTH or "wb_roi_width"¶

Description: Width of the area used for manual WB calculation. The sum of XI_PRM_WB_ROI_OFFSET_X and XI_PRM_WB_ROI_WIDTH must be equal or lower than the image resolution(width).

Supported cameras: MC,CB,MX

Type: Integer.
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_WB_ROI_WIDTH, &value);
xiSetParamInt(handle, XI_PRM_WB_ROI_WIDTH, value);
```

XI_PRM_WB_ROI_HEIGHT or "wb_roi_height"¶

Description: Height of the area used for manual WB calculation. The sum of XI_PRM_WB_ROI_OFFSET_Y and XI_PRM_WB_ROI_HEIGHT must be equal or lower than the image resolution(height).

Supported cameras: MC,CB,MX

Type: Integer.
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_WB_ROI_HEIGHT, &value);
xiSetParamInt(handle, XI_PRM_WB_ROI_HEIGHT, value);
```

XI_PRM_WB_KR or "wb_kr"¶

Description: White balance red coefficient.

Type: Float.
Default value: 1.0
Typical range: [0.0, 10.0]

Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_WB_KR, &value);
xiSetParamFloat(handle, XI_PRM_WB_KR, value);
```

XI_PRM_WB_KG or "wb_kg"¶

Description: White balance green coefficient.

Type: Float.
Default value: 1.0
Typical range: [0.0, 10.0]

Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_WB_KG, &value);
xiSetParamFloat(handle, XI_PRM_WB_KG, value);
```

XI_PRM_WB_KB or "wb_kb"¶

Description: White balance blue coefficient.

Type: Float.
Default value: 1.0
Typical range: [0.0, 10.0]

Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_WB_KB, &value);
xiSetParamFloat(handle, XI_PRM_WB_KB, value);
```

XI_PRM_WIDTH or "width"¶

Description: If camera runs in single region mode this parameter represents 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 value: Full resolution width.

Is invalidated by: XI_PRM_BINNING_HORIZONTAL, XI_PRM_DECIMATION_HORIZONTAL, XI_PRM_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING, XI_PRM_IMAGE_DATA_FORMAT, XI_PRM_IMAGE_AREA, XI_PRM_OUTPUT_DATA_PACKING, XI_PRM_OUTPUT_DATA_BIT_DEPTH, XI_PRM_HEIGHT
Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_WIDTH, &value);
xiSetParamInt(handle, XI_PRM_WIDTH, value);
```

XI_PRM_HEIGHT or "height"¶

Description: If camera runs in single region mode this parameter represents the 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 () this parameter is height of region currently selected.

Note1: Changing of this parameter will remove all images from buffer queue.

Note2: In case of using small ROI in combination with Fresco FL1100 controller, please read this article.

Type: Integer.
Default value: Full resolution width.

Is invalidated by: XI_PRM_BINNING_VERTICAL, XI_PRM_DECIMATION_VERTICAL, XI_PRM_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING, XI_PRM_IMAGE_DATA_FORMAT, XI_PRM_OUTPUT_DATA_PACKING, XI_PRM_IMAGE_AREA, XI_PRM_OUTPUT_DATA_BIT_DEPTH, XI_PRM_WIDTH
Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_HEIGHT, &value);
xiSetParamInt(handle, XI_PRM_HEIGHT, value);
```

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.

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

Type: Integer.
Default value: 0

Is invalidated by: XI_PRM_BINNING_HORIZONTAL, XI_PRM_DECIMATION_HORIZONTAL, XI_PRM_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING, XI_PRM_IMAGE_AREA
Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_OFFSET_X, &value);
xiSetParamInt(handle, XI_PRM_OFFSET_X, value);
```

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.

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

Type: Integer.
Default value: 0

Is invalidated by: XI_PRM_BINNING_VERTICAL, XI_PRM_DECIMATION_VERTICAL, XI_PRM_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING, XI_PRM_IMAGE_AREA
Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_OFFSET_Y, &value);
xiSetParamInt(handle, XI_PRM_OFFSET_Y, value);
```

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.

Note1: 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.

Type: Integer.
Default value: 0

Is invalidated by: XI_PRM_BINNING_VERTICAL, XI_PRM_BINNING_HORIZONTAL, XI_PRM_DECIMATION_VERTICAL, XI_PRM_DECIMATION_HORIZONTAL, XI_PRM_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING
Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_REGION_SELECTOR, &value);
xiSetParamInt(handle, XI_PRM_REGION_SELECTOR, value);
```

XI_PRM_REGION_MODE or "region_mode"¶
Description: Activates/deactivates Region selected by Region Selector in Multiple ROI

Note: Region 0 is always activated, it is not possible to deactivate it.

Type: Integer.
Default value: 1 for Region selector 0 and 0 for all other regions.
Typical range: [0, 1]

Is invalidated by: XI_PRM_BINNING_VERTICAL, XI_PRM_BINNING_HORIZONTAL, XI_PRM_DECIMATION_VERTICAL, XI_PRM_DECIMATION_HORIZONTAL,

XI_PRM_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_REGION_MODE, &value);
xiSetParamInt(handle, XI_PRM_REGION_MODE, value);
```

XI_PRM_HORIZONTAL_FLIP or "horizontal_flip"¶

Description: Activates horizontal flip if available in camera.

Type: Integer.
Default value: 0 for disabled flipping.

Usage:

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

XI_PRM_VERTICAL_FLIP or "vertical_flip"¶

Description: Activates vertical flip if available in camera.

Type: Integer.
Default value: XI_OFF

Usage:

```
xiSetParamInt(handle, XI_PRM_VERTICAL_FLIP, XI_ON);  //enable
vertical flipping
```

XI_PRM_INTERLINE_EXPOSURE_MODE or "interline_exposure_mode"¶

Description: Selector for Exposure parameter

Type: Enumerator.
Default value: XI_INTERLINE_EXPOSURE_MODE_OFF

Usage:

```
int interline_exposure_mode = 0;
xiGetParamInt(handle, XI_PRM_INTERLINE_EXPOSURE_MODE,
&interline_exposure_mode);
xiSetParamInt(handle, XI_PRM_INTERLINE_EXPOSURE_MODE,
XI_INTERLINE_EXPOSURE_MODE_OFF);
```

Value	Description
XI_INTERLINE_EXPOSURE_MODE_OFF	Disabled
XI_INTERLINE_EXPOSURE_MODE_ON	Enabled

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 value: XI_OFF

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_FFC, &value);
xiSetParamInt(handle, XI_PRM_FFC, XI_ON);
```

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.

Note: Use the same image file for this parameter as for XI_PRM_FFC_DARK_FIELD_FILE_NAME for dark-field correction only. Processing will subtract the dark image only while using the unity (1.00) gain for correction.

Type: String.
Default value: -

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 value: -

Usage:

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

XI_PRM_TOF_READOUT_MODE or "tof_readout_mode"¶
Description: Sets ToF Readout Mode

Type: Enumerator.
Default value: XI_TOF_READOUT_MODE_A_ONLY

Usage:

```
int tof_readout_mode = 0;
xiGetParamInt(handle, XI_PRM_TOF_READOUT_MODE, &tof_readout_mode);
xiSetParamInt(handle, XI_PRM_TOF_READOUT_MODE,
XI_TOF_READOUT_MODE_A_ONLY);
```

Value	Description
XI_TOF_READOUT_MODE_A_ONLY	A Only readout mode
XI_TOF_READOUT_MODE_B_ONLY	B Only readout mode
XI_TOF_READOUT_MODE_A_MINUS_B	A Minus B readout mode
XI_TOF_READOUT_MODE_A_PLUS_B	A Plus B readout mode
XI_TOF_READOUT_MODE_A_AND_B	A And B readout mode

XI_PRM_TOF_MODULATION_FREQUENCY or "tof_modulation_frequency"¶
Description: Sets ToF Modulation Frequency in MHz

Type: Float.
Default value: 100.0
Typical range: [4.0, 100.0]

Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_TOF_MODULATION_FREQUENCY, &value);
xiSetParamFloat(handle, XI_PRM_TOF_MODULATION_FREQUENCY, value);
```

XI_PRM_TOF_MULTIPLE_PHASES_IN_BUFFER or "tof_multiple_phases_in_buffer"¶
Description: is multiple ToF phases concatenated in buffer

Type: Integer.
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_TOF_MULTIPLE_PHASES_IN_BUFFER, &value);
```

XI_PRM_TOF_PHASES_COUNT or "tof_phases_count"¶

Description: Sets the number of tof phases. E.g. 4 for four phases.

Type: Integer.
Default value: 1

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_TOF_PHASES_COUNT, &value);
xiSetParamInt(handle, XI_PRM_TOF_PHASES_COUNT, value);
```

XI_PRM_TOF_PHASE_ANGLE or "tof_phase_angle"¶

Description: Sets Illumination angle for selected ToF phase

Type: Float.
Default value: 0.0
Typical range: [0.0, 360.0]

Is invalidated by: XI_PRM_TOF_PHASE_SELECTOR
Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_TOF_PHASE_ANGLE, &value);
xiSetParamFloat(handle, XI_PRM_TOF_PHASE_ANGLE, value);
```

XI_PRM_TOF_PHASE_EXPOSURE_TIME or "tof_phase_exposure_time"¶

Description: Sets Exposure time for selected ToF phase in microseconds.

Type: Float.
Default value: 1000.0

Is invalidated by: XI_PRM_TOF_PHASE_SELECTOR, XI_PRM_TOF_MODULATION_FREQUENCY
Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_TOF_PHASE_EXPOSURE_TIME, &value);
xiSetParamFloat(handle, XI_PRM_TOF_PHASE_EXPOSURE_TIME, value);
```

XI_PRM_TOF_PHASE_SELECTOR or "tof_phase_selector"¶
Description: Selects tof phase

Type: Integer.
Default value: 1

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_TOF_PHASE_SELECTOR, &value);
xiSetParamInt(handle, XI_PRM_TOF_PHASE_SELECTOR, value);
```

Image Format¶
Note: xiAPI allows to set different combinations of binning and decimation parameters.

On xiC, xiB, xiX, xiT cameras the parameters of units (Sensor, FPGA, CPU) are accessible with selectors (e.g. XI_PRM_BINNING_SELECTOR). After setting of selector, multiple parameters could be get of set for the selected unit. They can be divided into:

- Patterns (e.g. XI_PRM_BINNING_HORIZONTAL_PATTERN). If new pattern is set - the API might change the Values automatically in order to achieve setting of the new pattern.
- Values (e.g. XI_PRM_BINNING_HORIZONTAL). If new value is set - the API might change other values automatically in order to achieve setting of the new. Firstly it tries to find exact mode keeping the unchanged values, secondary it tries to find similar mode (trying to keep the other part - e.g. changing binning is trying to keep decimation parameters). If first and second attempts fails, the API tries to find any mode where new-value is found without keeping any other parameters, keeping Patterns.
- Modes for binning (e.g. XI_PRM_BINNING_VERTICAL_MODE)

XI_PRM_BINNING_SELECTOR or "binning_selector"¶
Description: Selects which binning engine is controlled by the BinningHorizontal and BinningVertical features.

Type: Enumerator.
Default value: XI_BIN_SELECT_SENSOR

Usage:

```
int binning_selector = 0;
xiGetParamInt(handle, XI_PRM_BINNING_SELECTOR, &binning_selector);
xiSetParamInt(handle, XI_PRM_BINNING_SELECTOR, XI_BIN_SELECT_SENSOR);
```

Value	Description
XI_BIN_SELECT_SENSOR	parameters for image sensor binning are selected
XI_BIN_SELECT_DEVICE_FPGA	parameters for device (camera) FPGA decimation are selected
XI_BIN_SELECT_HOST_CPU	parameters for Host CPU binning are selected

XI_PRM_BINNING_VERTICAL_MODE or "binning_vertical_mode"¶

Description: Sets the mode used to combine horizontal photo-sensitive cells together when BinningVertical is used.

Type: Enumerator.

Default value: XI_BIN_MODE_SUM

Usage:

```
int binning_vertical_mode = 0;
xiGetParamInt(handle, XI_PRM_BINNING_VERTICAL_MODE,
&binning_vertical_mode);
xiSetParamInt(handle, XI_PRM_BINNING_VERTICAL_MODE, XI_BIN_MODE_SUM);
```

Value	Description
XI_BIN_MODE_SUM	The response from the combined pixels will be added, resulting in increased sensitivity.
XI_BIN_MODE_AVERAGE	The response from the combined pixels will be averaged, resulting in increased signal/noise ratio.

XI_PRM_BINNING_VERTICAL or "binning_vertical"¶

Description: Number of vertical photo-sensitive cells to combine together. This reduces the vertical resolution (height) of the image.

Note: Setting this parameter may automatically change other Binning/Decimation parameters in order to achieve a valid combination.

Type: Integer.

Default value: 1

Typical range: The value range depends on camera model or associated selectors or invalidators.

Is invalidated by: XI_PRM_BINNING_SELECTOR, XI_PRM_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING, XI_PRM_SHUTTER_TYPE, XI_PRM_DECIMATION_VERTICAL, XI_PRM_DECIMATION_HORIZONTAL, XI_PRM_BINNING_HORIZONTAL, XI_PRM_DP_UNIT_SELECTOR, XI_PRM_DP_PROC_SELECTOR, XI_PRM_DP_PARAM_SELECTOR, XI_PRM_DP_PARAM_VALUE, XI_PRM_HDR

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_BINNING_VERTICAL, &value);
xiSetParamInt(handle, XI_PRM_BINNING_VERTICAL, value);
```

XI_PRM_BINNING_VERTICAL_FLOAT or "binning_vertical_float"¶
Description: Number of vertical photo-sensitive cells to combine together. This reduces the vertical resolution (height) of the image.

Note: Setting this parameter may automatically change other Binning/Decimation parameters in order to achieve a valid combination.

Type: Float.
Default value: 1.0
Typical range: [1.0, 4.0]

Is invalidated by: XI_PRM_BINNING_SELECTOR, XI_PRM_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING, XI_PRM_DECIMATION_VERTICAL, XI_PRM_DECIMATION_HORIZONTAL, XI_PRM_BINNING_HORIZONTAL, XI_PRM_DP_UNIT_SELECTOR, XI_PRM_DP_PROC_SELECTOR, XI_PRM_DP_PARAM_SELECTOR, XI_PRM_DP_PARAM_VALUE, XI_PRM_HDR
Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_BINNING_VERTICAL_FLOAT, &value);
xiSetParamFloat(handle, XI_PRM_BINNING_VERTICAL_FLOAT, value);
```

XI_PRM_BINNING_HORIZONTAL_MODE or "binning_horizontal_mode"¶
Description: Sets the mode to use to combine horizontal photo-sensitive cells together when BinningHorizontal is used.

Type: Enumerator.
Default value: XI_BIN_MODE_SUM

Usage:

```
int binning_horizontal_mode = 0;
xiGetParamInt(handle, XI_PRM_BINNING_HORIZONTAL_MODE,
&binning_horizontal_mode);
xiSetParamInt(handle, XI_PRM_BINNING_HORIZONTAL_MODE,
XI_BIN_MODE_SUM);
```

Value	Description
XI_BIN_MODE_SUM	The response from the combined pixels will be added, resulting in increased sensitivity.
XI_BIN_MODE_AVERAGE	The response from the combined pixels will be averaged, resulting in increased signal/noise ratio.

XI_PRM_BINNING_HORIZONTAL or "binning_horizontal"¶

Description: Number of horizontal photo-sensitive cells to combine together. This reduces the horizontal resolution (width) of the image.

Note: Setting this parameter may automatically change other Binning/Decimation parameters in order to achieve a valid combination.

Type: Integer.
Default value: 1
Typical range: The value range depends on camera model or associated selectors or invalidators.

Is invalidated by: XI_PRM_BINNING_SELECTOR, XI_PRM_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING, XI_PRM_SHUTTER_TYPE, XI_PRM_DECIMATION_VERTICAL, XI_PRM_DECIMATION_HORIZONTAL, XI_PRM_BINNING_VERTICAL, XI_PRM_DP_UNIT_SELECTOR, XI_PRM_DP_PROC_SELECTOR, XI_PRM_DP_PARAM_SELECTOR, XI_PRM_DP_PARAM_VALUE
Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_BINNING_HORIZONTAL, &value);
xiSetParamInt(handle, XI_PRM_BINNING_HORIZONTAL, value);
```

XI_PRM_BINNING_HORIZONTAL_FLOAT or "binning_horizontal_float"¶

Description: Number of horizontal photo-sensitive cells to combine together. This reduces the horizontal resolution (width) of the image.

Note: Setting this parameter may automatically change other Binning/Decimation parameters in order to achieve a valid combination.

Type: Float.
Default value: 1.0
Typical range: [1.0, 4.0]

Is invalidated by: XI_PRM_BINNING_SELECTOR, XI_PRM_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING, XI_PRM_DECIMATION_VERTICAL, XI_PRM_DECIMATION_HORIZONTAL, XI_PRM_BINNING_VERTICAL, XI_PRM_DP_UNIT_SELECTOR, XI_PRM_DP_PROC_SELECTOR, XI_PRM_DP_PARAM_SELECTOR, XI_PRM_DP_PARAM_VALUE

Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_BINNING_HORIZONTAL_FLOAT, &value);
xiSetParamFloat(handle, XI_PRM_BINNING_HORIZONTAL_FLOAT, value);
```

XI_PRM_BINNING_HORIZONTAL_PATTERN or "binning_horizontal_pattern"¶
Description: Defines number of horizontal photo-sensitive cells to combine.

Note: Setting this parameter may automatically change other Binning/Decimation parameters in order to achieve a valid combination.

Type: Enumerator.
Default value: XI_BIN_MONO

Usage:

```
int binning_horizontal_pattern = 0;
xiGetParamInt(handle, XI_PRM_BINNING_HORIZONTAL_PATTERN,
&binning_horizontal_pattern);
xiSetParamInt(handle, XI_PRM_BINNING_HORIZONTAL_PATTERN,
XI_BIN_MONO);
```

Value	Description
XI_BIN_MONO	adjacent pixels are combined
XI_BIN_BAYER	Bayer pattern is preserved during pixel combining

XI_PRM_BINNING_VERTICAL_PATTERN or "binning_vertical_pattern"¶
Description: Defines binning vertical pattern.

Note: Setting this parameter may automatically change other Binning/Decimation parameters in order to achieve a valid combination.

Type: Enumerator.
Default value: XI_BIN_MONO

Usage:

```
int binning_vertical_pattern = 0;
xiGetParamInt(handle, XI_PRM_BINNING_VERTICAL_PATTERN,
&binning_vertical_pattern);
xiSetParamInt(handle, XI_PRM_BINNING_VERTICAL_PATTERN, XI_BIN_MONO);
```

Value	Description
XI_BIN_MONO	adjacent pixels are combined
XI_BIN_BAYER	Bayer pattern is preserved during pixel combining

XI_PRM_DECIMATION_SELECTOR or "decimation_selector"¶
Description: Selects Decimation engine to configure.

Type: Enumerator.
Default value: XI_DEC_SELECT_SENSOR

Usage:

```
int decimation_selector = 0;
xiGetParamInt(handle, XI_PRM_DECIMATION_SELECTOR,
&decimation_selector);
xiSetParamInt(handle, XI_PRM_DECIMATION_SELECTOR,
XI_DEC_SELECT_SENSOR);
```

Value	Description
XI_DEC_SELECT_SENSOR	parameters for image sensor decimation are selected
XI_DEC_SELECT_DEVICE_FPGA	parameters for device (camera) FPGA decimation are selected
XI_DEC_SELECT_HOST_CPU	parameters for Host CPU decimation are selected

XI_PRM_DECIMATION_VERTICAL or "decimation_vertical"¶
Description: Vertical sub-sampling of the image. This reduces the vertical resolution (height) of the image by the specified vertical decimation factor.

Note: Setting this parameter may automatically change other Binning/Decimation parameters in order to achieve a valid combination.

Type: Integer.
Default value: 1
Typical range: The value range depends on camera model or associated selectors or invalidators.

Is invalidated by: XI_PRM_DECIMATION_VERTICAL, XI_PRM_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING, XI_PRM_BINNING_VERTICAL, XI_PRM_BINNING_HORIZONTAL, XI_PRM_DECIMATION_HORIZONTAL, XI_PRM_DP_UNIT_SELECTOR, XI_PRM_DP_PROC_SELECTOR, XI_PRM_DP_PARAM_SELECTOR, XI_PRM_DP_PARAM_VALUE, XI_PRM_DECIMATION_SELECTOR, XI_PRM_HDR
Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_DECIMATION_VERTICAL, &value);
xiSetParamInt(handle, XI_PRM_DECIMATION_VERTICAL, value);
```

XI_PRM_DECIMATION_HORIZONTAL or "decimation_horizontal"¶
Description: Horizontal sub-sampling of the image. This reduces the horizontal resolution (width) of the image by the specified horizontal decimation factor.

Note: Setting this parameter may automatically change other Binning/Decimation parameters in order to achieve a valid combination.

Type: Integer.
Default value: 1
Typical range: The value range depends on camera model or associated selectors or invalidators.

Is invalidated by: XI_PRM_DECIMATION_VERTICAL, XI_PRM_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING, XI_PRM_BINNING_VERTICAL, XI_PRM_BINNING_HORIZONTAL, XI_PRM_DECIMATION_VERTICAL, XI_PRM_DP_UNIT_SELECTOR, XI_PRM_DP_PROC_SELECTOR, XI_PRM_DP_PARAM_SELECTOR, XI_PRM_DP_PARAM_VALUE, XI_PRM_DECIMATION_SELECTOR
Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_DECIMATION_HORIZONTAL, &value);
xiSetParamInt(handle, XI_PRM_DECIMATION_HORIZONTAL, value);
```

XI_PRM_DECIMATION_HORIZONTAL_PATTERN or "decimation_horizontal_pattern"¶
Description: Defines decimation horizontal pattern.

Note: Setting this parameter may automatically change other Binning/Decimation parameters in order to achieve a valid combination.

Type: Enumerator.
Default value: XI_DEC_MONO

Usage:

```
int decimation_horizontal_pattern = 0;
xiGetParamInt(handle, XI_PRM_DECIMATION_HORIZONTAL_PATTERN,
&decimation_horizontal_pattern);
xiSetParamInt(handle, XI_PRM_DECIMATION_HORIZONTAL_PATTERN,
XI_DEC_MONO);
```

Value	Description

XI_DEC_MONO	adjacent pixels are decimated
XI_DEC_BAYER	Bayer pattern is preserved during pixel decimation

XI_PRM_DECIMATION_VERTICAL_PATTERN or "decimation_vertical_pattern"¶
Description: Defines decimation vertical pattern.

Note: Setting this parameter may automatically change other Binning/Decimation parameters in order to achieve a valid combination.

Type: Enumerator.
Default value: XI_DEC_MONO

Usage:

```
int decimation_vertical_pattern = 0;
xiGetParamInt(handle, XI_PRM_DECIMATION_VERTICAL_PATTERN,
&decimation_vertical_pattern);
xiSetParamInt(handle, XI_PRM_DECIMATION_VERTICAL_PATTERN,
XI_DEC_MONO);
```

Value	Description
XI_DEC_MONO	adjacent pixels are decimated
XI_DEC_BAYER	Bayer pattern is preserved during pixel decimation

---

AE Setup¶

XI_PRM_EXP_PRIORITY or "exp_priority"¶
Description: Exposure priority for Auto Exposure / Auto Gain function.

- Value: **1.0** >>> meaning: **Exposure priority. Only exposure will be changed.**
- Value: **0.5** >>> meaning: **Exposure and gain will be used (50%:50%).**
- Value: **0.0** >>> meaning: **Gain priority. Only gain will be changed.**

Type: Float.
Default value: 1.0
Typical range: [0.0, 1.0]

Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_EXP_PRIORITY, &value);
xiSetParamFloat(handle, XI_PRM_EXP_PRIORITY, value);
```

XI_PRM_AG_MAX_LIMIT or "ag_max_limit"¶

Description: Maximum limit of gain in AEAG procedure.

Type: Float.
Default value: Depends on camera type (dB).

Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_AG_MAX_LIMIT, &value);
xiSetParamFloat(handle, XI_PRM_AG_MAX_LIMIT, value);
```

XI_PRM_AE_MAX_LIMIT or "ae_max_limit"¶

Description: Maximum limit of exposure (in uSec) in AEAG procedure.

Type: Integer.
Default value: 200000
Typical range: [0, 1000000]

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_AE_MAX_LIMIT, &value);
xiSetParamInt(handle, XI_PRM_AE_MAX_LIMIT, value);
```

XI_PRM_AEAG_LEVEL or "aeag_level"¶

Description: Average intensity of output signal AEAG should achieve(in %).

Type: Integer.
Default value: 50
Typical range: [0, 100]

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_AEAG_LEVEL, &value);
xiSetParamInt(handle, XI_PRM_AEAG_LEVEL, value);
```

Performance¶

XI_PRM_LIMIT_BANDWIDTH or "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 by 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.

Note: Controlling method (clock or line period) depends on the camera model.

Type: Integer.
Default value: 0

Usage:

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

Example: See more at our application note about Multiple Cameras Setup.

```
// get interface data rate
int interface_data_rate_mbps=2500;
// calculate data rate 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);
```

XI_PRM_LIMIT_BANDWIDTH_MODE or "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.

Note: This parameter is not supported on MQ, MU, MD, MR camera families.

Type: Enumerator.
Default value: XI_ON

Usage:

```
int limit_bandwidth_mode = 0;
xiGetParamInt(handle, XI_PRM_LIMIT_BANDWIDTH_MODE,
&limit_bandwidth_mode);
xiSetParamInt(handle, XI_PRM_LIMIT_BANDWIDTH_MODE, XI_OFF);
```

Value	Description
XI_OFF	Turn parameter off
XI_ON	Turn parameter on

XI_PRM_SENSOR_DATA_BIT_DEPTH or "sensor_bit_depth"¶
Description: Returns the bit depth of the pixel data received from sensor.

Note: Read more at XiAPI Image Data Flow.

Type: Enumerator.
Default value: 0

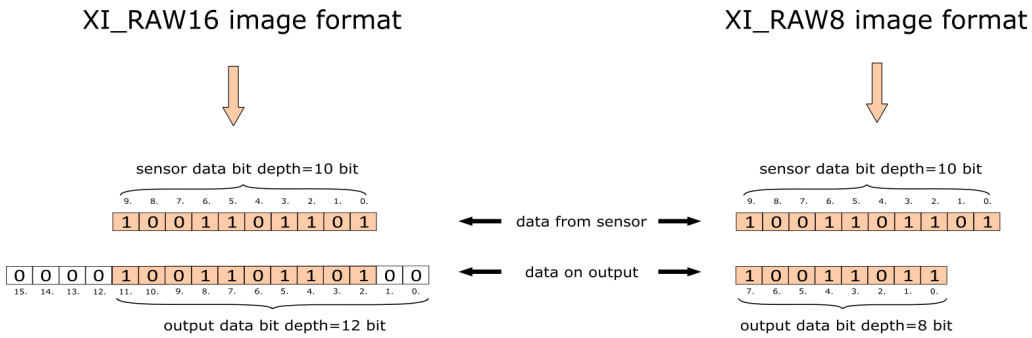
Is invalidated by: XI_PRM_IMAGE_DATA_FORMAT, XI_PRM_USER_SET_LOAD, XI_PRM_DUAL_ADC_MODE, XI_PRM_DOWNSAMPLING
Usage:

```
int sensor_bit_depth = 0;
xiGetParamInt(handle, XI_PRM_SENSOR_DATA_BIT_DEPTH,
&sensor_bit_depth);
xiSetParamInt(handle, XI_PRM_SENSOR_DATA_BIT_DEPTH, XI_BPP_8);
```

Value	Description
XI_BPP_8	8 bit per pixel
XI_BPP_9	9 bit per pixel
XI_BPP_10	10 bit per pixel
XI_BPP_11	11 bit per pixel
XI_BPP_12	12 bit per pixel
XI_BPP_13	13 bit per pixel
XI_BPP_14	14 bit per pixel
XI_BPP_15	15 bit per pixel
XI_BPP_16	16 bit per pixel
XI_BPP_24	24 bit per pixel
XI_BPP_32	32 bit per pixel

XI_PRM_OUTPUT_DATA_BIT_DEPTH or "output_bit_depth"¶
Description: The bit depth of the output data from camera (=transport layer).

Note: Read more at XiAPI Image Data Flow.



Type: Enumerator.
Default value: 0

Is invalidated by: XI_PRM_IMAGE_DATA_FORMAT, XI_PRM_DP_PARAM_VALUE, XI_PRM_BINNING_VERTICAL, XI_PRM_BINNING_HORIZONTAL, XI_PRM_DECIMATION_VERTICAL, XI_PRM_DECIMATION_HORIZONTAL, XI_PRM_SHUTTER_TYPE
Usage:

```
int output_bit_depth = 0;
xiGetParamInt(handle, XI_PRM_OUTPUT_DATA_BIT_DEPTH,
&output_bit_depth);
xiSetParamInt(handle, XI_PRM_OUTPUT_DATA_BIT_DEPTH, XI_BPP_8);
```

Value	Description
XI_BPP_8	8 bit per pixel
XI_BPP_9	9 bit per pixel
XI_BPP_10	10 bit per pixel
XI_BPP_11	11 bit per pixel
XI_BPP_12	12 bit per pixel
XI_BPP_13	13 bit per pixel
XI_BPP_14	14 bit per pixel
XI_BPP_15	15 bit per pixel
XI_BPP_16	16 bit per pixel

XI_BPP_24	24 bit per pixel
XI_BPP_32	32 bit per pixel

XI_PRM_IMAGE_DATA_BIT_DEPTH or "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 the alignment of the data on the xiGetImage.

Type: Enumerator.
Default value: 0

Is invalidated by: XI_PRM_IMAGE_DATA_FORMAT
Usage:

```
int image_data_bit_depth = 0;
xiGetParamInt(handle, XI_PRM_IMAGE_DATA_BIT_DEPTH,
&image_data_bit_depth);
xiSetParamInt(handle, XI_PRM_IMAGE_DATA_BIT_DEPTH, XI_BPP_8);
```

Value	Description
XI_BPP_8	8 bit per pixel
XI_BPP_9	9 bit per pixel
XI_BPP_10	10 bit per pixel
XI_BPP_11	11 bit per pixel
XI_BPP_12	12 bit per pixel
XI_BPP_13	13 bit per pixel
XI_BPP_14	14 bit per pixel
XI_BPP_15	15 bit per pixel
XI_BPP_16	16 bit per pixel
XI_BPP_24	24 bit per pixel
XI_BPP_32	32 bit per pixel

XI_PRM_OUTPUT_DATA_PACKING or "output_bit_packing"¶

Description: This feature enables bit packing on transport data layer, thus increasing the maximum frame rate when data with 10 or 12 bits per pixel is transported. For more info please see Transport Data Packing feature description.

Note: Read more at XiAPI Image Data Flow.

Type: Integer.
Default value: XI_OFF

Is invalidated by: XI_PRM_IMAGE_DATA_FORMAT, XI_PRM_OUTPUT_DATA_BIT_DEPTH, XI_PRM_DP_PARAM_VALUE, XI_PRM_BINNING_VERTICAL, XI_PRM_BINNING_HORIZONTAL, XI_PRM_DECIMATION_VERTICAL, XI_PRM_DECIMATION_HORIZONTAL, XI_PRM_SHUTTER_TYPE
Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_OUTPUT_DATA_PACKING, &value);
xiSetParamInt(handle, XI_PRM_OUTPUT_DATA_PACKING, XI_ON);
```

XI_PRM_OUTPUT_DATA_PACKING_TYPE or "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: Enumerator.
Default value: XI_DATA_PACK_XI_GROUPING

Usage:

```
int output_bit_packing_type = 0;
xiGetParamInt(handle, XI_PRM_OUTPUT_DATA_PACKING_TYPE,
&output_bit_packing_type);
xiSetParamInt(handle, XI_PRM_OUTPUT_DATA_PACKING_TYPE,
XI_DATA_PACK_XI_GROUPING);
```

Value	Description
XI_DATA_PACK_XI_GROUPING	Data grouping (10g160, 12g192, 14g224).
XI_DATA_PACK_PFNC_LSB_PACKING	Data packing (10p, 12p)

Temperature¶

XI_PRM_IS_COOLED or "iscooled"¶
Description: Returns 1 for cameras that support cooling.

Type: Integer.
Default value: XI_OFF

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_IS_COOLED, &value);
```

XI_PRM_COOLING or "cooling"¶

Description: Set camera cooling control. Replaced by XI_PRM_TEMP_CONTROL_MODE

Type: Enumerator.
Default value: XI_TEMP_CTRL_MODE_OFF

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);
```

Value	Description
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 parameter TARGET_TEMP.
XI_TEMP_CTRL_MODE_MANUAL	Controlling of elements is done manually by application.

XI_PRM_TARGET_TEMP or "target_temp"¶

Description: Set target temperature for automatic temperature control.

Type: Float.
Default value: 0.0
Typical range: [0.0, 200.0]

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: Enumerator.
Default value: XI_TEMP_SENSOR_BOARD

Usage:

```
float temperature = 0.0;
```

```
xiSetParamInt(handle, XI_PRM_TEMP_SELECTOR,
XI_TEMP_IMAGE_SENSOR_DIE_RAW);
xiGetParamFloat(handle, XI_PRM_TEMP, &temperature);
```

Value	Description
XI_TEMP_IMAGE_SENSOR_DIE_RAW	Image sensor die (non-calibrated)
XI_TEMP_IMAGE_SENSOR_DIE	Image sensor die (calibrated)
XI_TEMP_SENSOR_BOARD	Image sensor PCB
XI_TEMP_INTERFACE_BOARD	Data interface PCB
XI_TEMP_FRONT_HOUSING	Front part of camera housing
XI_TEMP_REAR_HOUSING	Rear part of camera housing
XI_TEMP_TEC1_COLD	TEC1 cold side temperature
XI_TEMP_TEC1_HOT	TEC1 hot side temperature
XI_TEMP_VCSEL_BOARD_A	VCSEL board 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 value: 0.0

Is invalidated by: XI_PRM_TEMP_SELECTOR
Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_TEMP, &value);
```

XI_PRM_TEMP_CONTROL_MODE or "device_temperature_ctrl_mode"¶
Description: Sets temperature control mode.

Note: On some camera models, when some component (e.g. housing) reaches critical temperature, the mode is changed to XI_TEMP_CTRL_MODE_OFF automatically by camera and this mode remains off. It can be re-enabled by setting mode to XI_TEMP_CTRL_MODE_AUTO. By getting XI_PRM_TEMP_CONTROL_MODE, application can get the information, about current state.

Type: Enumerator.
Default value: XI_TEMP_CTRL_MODE_OFF

Usage:

```
xiSetParamFloat(handle, XI_PRM_TARGET_TEMP, 18.5);
xiSetParamInt(handle, XI_PRM_TEMP_CONTROL_MODE,
XI_TEMP_CTRL_MODE_AUTO);
// check the current mode periodically
xiGetParamInt(handle, XI_PRM_TEMP_CONTROL_MODE, &control_mode);
if (XI_TEMP_CTRL_MODE_AUTO == control_mode)
   printf("Temperature is controlled automatically.");
```

Value	Description
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 parameter TARGET_TEMP.
XI_TEMP_CTRL_MODE_MANUAL	Controlling of elements is done manually by application.

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 value: 0.0

Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_CHIP_TEMP, &value);
```

XI_PRM_HOUS_TEMP or "hous_temp"¶

Description: Camera housing temperature.

Type: Float.
Default value: 0.0

Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_HOUS_TEMP, &value);
```

XI_PRM_HOUS_BACK_SIDE_TEMP or "hous_back_side_temp"¶

Description: Camera housing back side temperature.

Type: Float.
Default value: 0.0

Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_HOUS_BACK_SIDE_TEMP, &value);
```

XI_PRM_SENSOR_BOARD_TEMP or "sensor_board_temp"¶
Description: Camera sensor board temperature.

Type: Float.
Default value: 0.0

Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_SENSOR_BOARD_TEMP, &value);
```

XI_PRM_TEMP_ELEMENT_SEL or "device_temperature_element_sel"¶
Description: Temperature element selector (TEC, Fan)

Type: Enumerator.
Default value: XI_TEMP_ELEM_TEC1

Usage:

See `XI_PRM_TEMP_ELEMENT_VALUE`

Value	Description
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	Temperature element fan current or rotation (FAN1 = Fan)
XI_TEMP_ELEM_FAN1_THRS_TEMP	Temperature element fan start rotation threshold temperature

XI_PRM_TEMP_ELEMENT_VALUE or "device_temperature_element_val"¶
Description: Temperature element value in percents of full control range.

Type: Float.
Default value: 0.0
Typical range: [0.0, 100.0]

Is invalidated by: XI_PRM_TEMP_ELEMENT_SEL

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);
```

Color Correction¶

Note: Works only for color cameras.

XI_PRM_CMS or "cms"¶

Description: Enable or disable color management.

Note: This feature is in Beta stage.

Type: Enumerator.
Default value: XI_CMS_DIS

Usage:

```
int cms = 0;
xiGetParamInt(handle, XI_PRM_CMS, &cms);
xiSetParamInt(handle, XI_PRM_CMS, XI_CMS_DIS);
```

Value	Description
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)

XI_PRM_CMS_INTENT or "cms_intent"¶

Description: Defines rendering intents. See more at our support page CMS INTENT.

Note1: This feature is in Beta stage.

Type: Enumerator.
Default value: XI_CMS_INTENT_PERCEPTUAL

Usage:

```
int cms_intent = 0;
xiGetParamInt(handle, XI_PRM_CMS_INTENT, &cms_intent);
xiSetParamInt(handle, XI_PRM_CMS_INTENT, XI_CMS_INTENT_PERCEPTUAL);
```

Value	Description
XI_CMS_INTENT_PERCEPTUAL	CMS intent perceptual
XI_CMS_INTENT_RELATIVE_COLORIMETRIC	CMS intent relative colorimetry
XI_CMS_INTENT_SATURATION	CMS intent saturation
XI_CMS_INTENT_ABSOLUTE_COLORIMETRIC	CMS intent absolute colorimetry

XI_PRM_APPLY_CMS or "apply_cms"¶
Description: If set to XI_ON applies CMS profile to xiGetImage.

Type: Integer.
Default value: XI_OFF

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_APPLY_CMS, &value);
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 value: (default device profile)

Usage:

```
xiSetParamString(handle, 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 value: (sRGB profile)

Usage:

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

XI_PRM_IMAGE_IS_COLOR or "iscolor"¶

Description: Returns 1 for color cameras.

Type: Integer.
Default value: XI_OFF

Usage:

```
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: Enumerator.
Default value: XI_CFA_NONE

Usage:

```
int cfa = 0;
xiGetParamInt(handle, XI_PRM_COLOR_FILTER_ARRAY, &cfasizeof(value));
```

Value	Description
XI_CFA_NONE	Result pixels have no filters applied in this format
XI_CFA_BAYER_RGGB	Regular RGGB
XI_CFA_CMYG	AK Sony sens
XI_CFA_RGR	2R+G readout
XI_CFA_BAYER_BGGR	BGGR readout
XI_CFA_BAYER_GRBG	GRBG readout
XI_CFA_BAYER_GBRG	GBRG readout
XI_CFA_POLAR_A_BAYER_BGGR	BGGR polarized 4x4 macropixel
XI_CFA_POLAR_A	Polarized 2x2 macropixel
XI_CFA_TOF_ANB	ToF A and B
XI_CFA_TOF_AMB	ToF A minus B
XI_CFA_TOF_APB	ToF A plus B
XI_CFA_TOF_A	ToF A only

XI_CFA_TOF_B	ToF B only

XI_PRM_GAMMAY or "gammaY"¶

Description: Luminosity gamma.

Lowering the value increases correction.

Type: Float.
Default value: 0.47
Typical range: [0.3, 1.0]

Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_GAMMAY, &value);
xiSetParamFloat(handle, XI_PRM_GAMMAY, value);
```

XI_PRM_GAMMAC or "gammaC"¶

Description: Chromaticity gamma.

Type: Float.
Default value: 0.8
Typical range: [0.0, 1.0]

Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_GAMMAC, &value);
xiSetParamFloat(handle, XI_PRM_GAMMAC, value);
```

XI_PRM_SHARPNESS or "sharpness"¶

Description: Sharpness Strength. Increasing the value results in sharper image.

Note: Works also for XI_MONO* formats, but only for color cameras.

Type: Float.
Default value: 0.0
Typical range: [-4.0, 4.0]

Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_SHARPNESS, &value);
xiSetParamFloat(handle, XI_PRM_SHARPNESS, value);
```

XI_PRM_CC_MATRIX_00 or "ccMTX00"¶

Description: Color Correction Matrix element [0][0].

Correction Matrix elements:

```
        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
```

Type: Float.

Default value: 1.0

Typical range: [-8.0, 8.0]

Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_CC_MATRIX_00, &value);
xiSetParamFloat(handle, XI_PRM_CC_MATRIX_00, value);
```

XI_PRM_DEFAULT_CC_MATRIX or "defccMTX"¶

Description: Set default Color Correction Matrix

Type: Integer.

Default value: 0

Usage:

```
int value = 0;
xiSetParamInt(handle, XI_PRM_DEFAULT_CC_MATRIX, value);
```

XI_PRM_CC_MATRIX_NORM or "ccMTXnorm"¶

Description: Activates normalization of color correction matrix.

Type: Integer.

Default value: 0 for disabled normalization.

Usage:

```
xiSetParamInt(handle, XI_PRM_CC_MATRIX_NORM, XI_ON);  //enable color
```

```
correction matrix normalization
```

Device IO¶

XI_PRM_TRG_SOURCE or "trigger_source"¶

Description: Defines source of trigger.

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.

Type: Enumerator.

Default value: XI_TRG_OFF

Usage:

```
int trigger_source = 0;
xiGetParamInt(handle, XI_PRM_TRG_SOURCE, &trigger_source);
xiSetParamInt(handle, XI_PRM_TRG_SOURCE, XI_TRG_OFF);
```

Value	Description
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.
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.

Example: For more examples see xiAPI Camera Trigger and Synchronization Signals.

```
// 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;
                // returns error in case if it is too early
                // to start next exposure
                Sleep(1);
                }
                // get the image
        xiGetImage(handle, 1000, &image);
        }
```

XI_PRM_TRG_SOFTWARE or "trigger_software"¶

Description: Generates an internal trigger. XI_PRM_TRG_SOURCE has to be set to XI_TRG_SOFTWARE.

Note: Some models (xiMU - MU9 and xiQ) return error code if sensor is not ready to start exposure of next image. Other cameras return XI_OK even if sensor is not ready to start exposure.

Type: Integer.
Default value: 0

Usage:

```
int value = 0;
xiSetParamInt(handle, XI_PRM_TRG_SOFTWARE, value);
```

XI_PRM_TRG_SELECTOR or "trigger_selector"¶

Description: This parameter selects the type of trigger. For more information about enumerator XI_TRG_SEL_EXPOSURE_ACTIVE please refer to our Exposure Defined by Trigger Pulse Length support page.

For more information about enumerators: XI_TRG_SEL_FRAME_BURST_START, XI_TRG_SEL_FRAME_BURST_ACTIVE please refer to our Frame Burst Modes support page.

For more information about enumerator XI_TRG_SEL_EXPOSURE_START please refer to our Multiple exposures in one frame support page.

Type: Enumerator.
Default value: XI_TRG_SEL_FRAME_START

Usage:

```
int trigger_selector = 0;
```

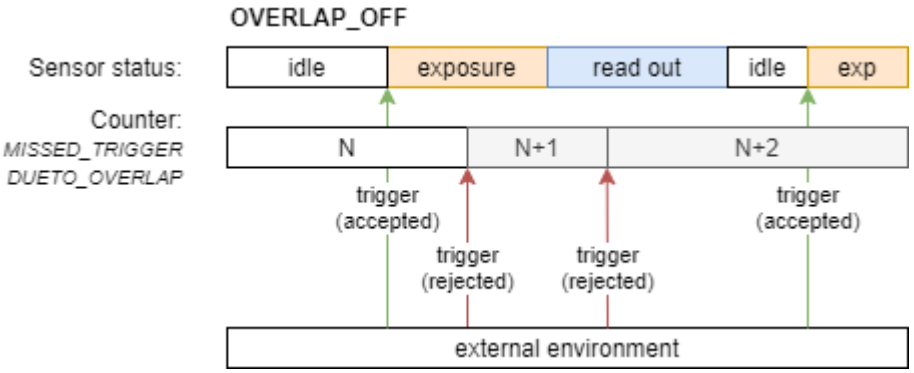
```
xiGetParamInt(handle, XI_PRM_TRG_SELECTOR, &trigger_selector);
xiSetParamInt(handle, XI_PRM_TRG_SELECTOR, XI_TRG_SEL_FRAME_START);
```

Value	Description
XI_TRG_SEL_FRAME_START	Trigger starts the capture of one frame
XI_TRG_SEL_EXPOSURE_ACTIVE	Trigger controls the start and length of the exposure.
XI_TRG_SEL_FRAME_BURST_START	Trigger starts the capture of the bursts of frames in an acquisition.
XI_TRG_SEL_FRAME_BURST_ACTIVE	Trigger controls the duration of the capture of the bursts of frames in an acquisition.
XI_TRG_SEL_MULTIPLE_EXPOSURES	Trigger which when first trigger starts exposure and consequent pulses are gating exposure(active HI)
XI_TRG_SEL_EXPOSURE_START	Trigger controls the start of the exposure of one Frame.
XI_TRG_SEL_MULTI_SLOPE_PHASE_CHANGE	Trigger controls the multi slope phase in one Frame (phase0 -> phase1) or (phase1 -> phase2).
XI_TRG_SEL_ACQUISITION_START	Trigger starts acquisition of first frame.

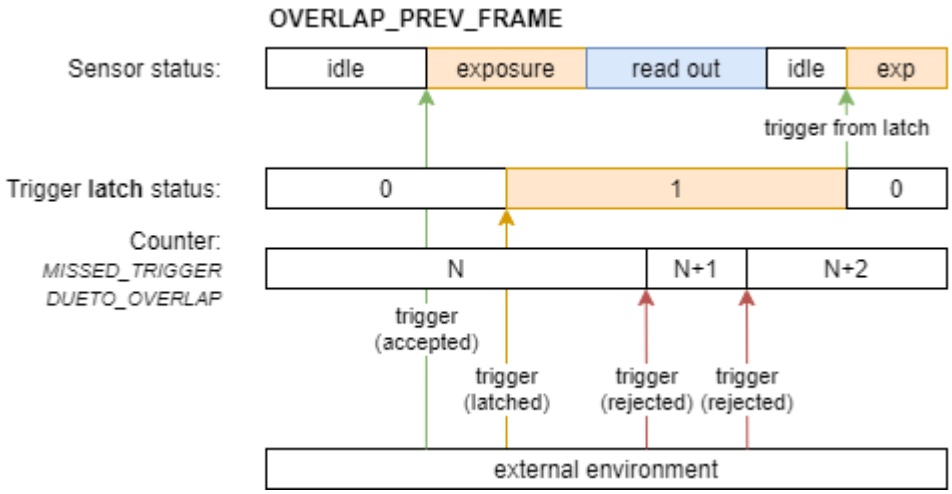
XI_PRM_TRG_OVERLAP or "trigger_overlap"¶

Description: Specifies the type of trigger overlap permitted with the previous frame. This defines when a valid trigger will be accepted (or latched) for a new frame.

In XI_TRG_OVERLAP_OFF - no trigger overlap is permitted. If camera is in read-out phase, all triggers are rejected.



In XI_TRG_OVERLAP_PREV_FRAME - trigger is accepted by camera any time. If sensor is not ready for the next exposure - the trigger is latched and sensor starts exposure as soon as exposure can be started with defined exposure time.



Type: Enumerator.

Default value: XI_TRG_OVERLAP_PREV_FRAME

Is invalidated by: XI_PRM_TRG_SELECTOR, XI_PRM_EXPOSURE_BURST_COUNT

Usage:

```
int trigger_overlap = 0;
xiGetParamInt(handle, XI_PRM_TRG_OVERLAP, &trigger_overlap);
xiSetParamInt(handle, XI_PRM_TRG_OVERLAP, XI_TRG_OVERLAP_OFF);
```

Value	Description
XI_TRG_OVERLAP_OFF	No trigger overlap is permitted. If camera is in read-out phase, all triggers are rejected.
XI_TRG_OVERLAP_READ_OUT	Trigger is accepted only when sensor is ready to start next exposure with defined exposure time. Trigger is rejected when sensor is not ready for new exposure with defined exposure time. (see Note1)
XI_TRG_OVERLAP_PREV_FRAME	Trigger is accepted by camera any time. If sensor is not ready for the next exposure - the trigger is latched and sensor starts exposure as soon as exposure can be started with defined exposure time.

Note1: This mode is planned and not yet supported by cameras.

XI_PRM_ACQ_FRAME_BURST_COUNT or "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: Integer.

Default value: 1

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_ACQ_FRAME_BURST_COUNT, &value);
xiSetParamInt(handle, XI_PRM_ACQ_FRAME_BURST_COUNT, value);
```

XI_PRM_TIMESTAMP or "timestamp"¶

Description: Reads the current timestamp value from camera in nanoseconds (only valid for xiB, xiC, xiX camera families).

Type: Unsigned integer 64 bit.
Default value: 0

Usage:

```
uint64_t value = 0;
DWORD size = sizeof(value);
XI_PRM_TYPE type = xiTypeInteger64;
xiGetParam(handle, XI_PRM_TIMESTAMP, &value, &size, &type);
```

GPIO Setup¶

XI_PRM_GPI_SELECTOR or "gpi_selector"¶

Description: Selects GPI.

Type: Enumerator.
Default value: 1

Usage:

```
int gpi_selector = 0;
xiGetParamInt(handle, XI_PRM_GPI_SELECTOR, &gpi_selector);
xiSetParamInt(handle, XI_PRM_GPI_SELECTOR, XI_GPI_PORT1);
```

Value	Description
XI_GPI_PORT1	GPI port 1
XI_GPI_PORT2	GPI port 2
XI_GPI_PORT3	GPI port 3
XI_GPI_PORT4	GPI port 4
XI_GPI_PORT5	GPI port 5

XI_GPI_PORT6	GPI port 6
XI_GPI_PORT7	GPI port 7
XI_GPI_PORT8	GPI port 8
XI_GPI_PORT9	GPI port 9
XI_GPI_PORT10	GPI port 10
XI_GPI_PORT11	GPI port 11
XI_GPI_PORT12	GPI port 12

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.

Models: MQ-S7 cameras (MQ.*-S7)

gpi_index	physical pin on the camera	cable color
1	3 (IN1 - optical)	Blue

Models: MQ cameras (MQ.*)

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

Models: MJ cameras (MJ.*)

gpi_index	physical pin on the camera
1	3 (IN1 - optical)

Models: MC -UC cameras (MC.*-UC)

gpi_index	physical pin on the camera
1	11 (INOUT1)
2	3 (INOUT2)

Models: MC -FL/FV cameras (MC.*-F.*)

gpi_index	physical pin on the camera
1	15 (IN1 - optical)

Models: MC -UB,-TC cameras (MC.*)

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

Models: MU181CR-ON camera (MU181CR-.*)

gpi_index	physical pin on the camera
1	1 (GX1)
2	3 (GX2)
3	5 (GX3)

Models: MU TOF cameras (MU.*TG.*-UC)

gpi_index	physical pin on the camera
1	11 (INOUT1)
2	3 (INOUT2)

Models: MU cameras (MU.*-.*)

gpi_index	physical pin on the camera
1	1 (GX1)
2	3 (GX2)
3	5 (GX3)
4	7 (GX4)

Models: MX X2G2 cameras (MX.*X2G2.*)

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

Models: MX X4G2 cameras (MX.*X4G2.*)

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: MX X4G3 cameras (MX.*X4G3.*)

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)

Models: MX X8G3 cameras (MX.*X8G3-FF.*)

gpi_index	**physical pin on the camera**
1	IO-A 1 (IN1 - optical)
2	IO-A 3 (IN2 - optical)
3	IO-B 1 (INOUT1)
4	IO-B 3 (INOUT2)
5	IO-B 4 (INOUT3)
6	IO-B 6 (INOUT4)

Models: CB X8G3 cameras (CB.*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)

Models: CB X4G2 cameras (CB.*)

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: MX377 MTP cameras (MX377.*MTP.*)

gpi_index	physical pin on the camera
2	8 (IN2 - optical)
3	13 (IN3 - optical)
4	2 (INOUT1)
5	3 (INOUT2)
6	4 (INOUT3)
7	5 (INOUT4)
8	12 (INOUT5)
9	14 (INOUT6)
10	9 (INOUT7)
11	7 (INOUT8)

XI_PRM_GPI_MODE or "gpi_mode"¶
Description: Defines GPI functionality.

Note1: 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

Note2: If bidirectional (input/output) pin is used, set XI_PRM_GPO_MODE to XI_GPO_HIGH_IMPEDANCE. This will disable output driver on the pin from camera.

Type: Enumerator.
Default value: XI_GPI_OFF

Is invalidated by: XI_PRM_GPI_SELECTOR
Usage:

```
int gpi_mode = 0;
xiGetParamInt(handle, XI_PRM_GPI_MODE, &gpi_mode);
xiSetParamInt(handle, XI_PRM_GPI_MODE, XI_GPI_OFF);
```

Value	Description
XI_GPI_OFF	Input is not used for triggering, but can be used to get parameter 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)

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);
```

XI_PRM_GPI_LEVEL or "gpi_level"¶
Description: Level of digital input selected by XI_PRM_GPI_SELECTOR.

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.

Type: Integer.
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_GPI_LEVEL, &value);
```

Example:

```
// select digital input (different mapping to physical pin on each
model, see table below)
int gpi_index = 1;
xiSetParamInt(handle, XI_PRM_GPI_SELECTOR,  gpi_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", gpi_index, gpi_level);
```

XI_PRM_GPI_LEVEL_AT_IMAGE_EXP_START or "gpi_level_at_image_exp_start"¶

Description: Level of digital input selected by XI_PRM_GPI_SELECTOR sampled at exposure start of the last image received by GetImage.

Type: Integer.
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_GPI_LEVEL_AT_IMAGE_EXP_START, &value);
```

Example:

```
if (XI_OK == xiGetImage(xiH, 5000, &image))
{
  int level_input_1_at_exp_start = 0;
  xiSetParamInt(xiH, XI_PRM_GPI_SELECTOR, 1);
  xiGetParamInt(xiH,XI_PRM_GPI_LEVEL_AT_IMAGE_EXP_START,
&level_input_1_at_exp_start);
}
```

XI_PRM_GPI_LEVEL_AT_IMAGE_EXP_END or "gpi_level_at_image_exp_end"¶

Description: Level of digital input selected by XI_PRM_GPI_SELECTOR sampled at exposure end of the last image received by GetImage.

Type: Integer.
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_GPI_LEVEL_AT_IMAGE_EXP_END, &value);
```

Example:

```
if (XI_OK == xiGetImage(xiH, 5000, &image))
{
  int level_input_2_at_exp_end = 0;
  xiSetParamInt(xiH, XI_PRM_GPI_SELECTOR, 2);
  xiGetParamInt(xiH, XI_PRM_GPI_LEVEL_AT_IMAGE_EXP_END,
&level_input_2_at_exp_end);
}
```

XI_PRM_GPO_SELECTOR or "gpo_selector"¶

Description: Selects GPO.

Type: Enumerator.

Default value: 1

Usage:

```
int gpo_selector = 0;
xiGetParamInt(handle, XI_PRM_GPO_SELECTOR, &gpo_selector);
xiSetParamInt(handle, XI_PRM_GPO_SELECTOR, XI_GPO_PORT1);
```

Value	Description
XI_GPO_PORT1	GPO port 1
XI_GPO_PORT2	GPO port 2
XI_GPO_PORT3	GPO port 3
XI_GPO_PORT4	GPO port 4
XI_GPO_PORT5	GPO port 5
XI_GPO_PORT6	GPO port 6
XI_GPO_PORT7	GPO port 7
XI_GPO_PORT8	GPO port 8
XI_GPO_PORT9	GPO port 9
XI_GPO_PORT10	GPO port 10
XI_GPO_PORT11	GPO port 11
XI_GPO_PORT12	GPO port 12

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.

Models: MQ-S7 cameras (MQ.*-S7)

gpo_index	physical pin on the camera	cable color
1	4 (OUT1 - optical)	Black

Models: MQ cameras (MQ.*)

gpo_index	physical pin on the camera	cable color
1	3 (OUT1 - optical)	White

Models: MJ cameras (MJ.*)

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

Models: MC -UC cameras (MC.*-UC)

gpo_index	physical pin on the camera
2	12 (OUT1)
3	11 (INOUT1)
1	4 (OUT2)
4	3 (INOUT2)

Models: MC -FL/FV cameras (MC.*-F.*)

gpo_index	physical pin on the camera
1	13 (OUT1 - optical)

Models: MC -UB,-TC cameras (MC.*)

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

Models: MU181CR-ON camera (MU181CR-.*)

gpo_index	physical pin on the camera
1	1 (GX1)

2	3 (GX2)
3	5 (GX3)

Models: MU TOF cameras (MU.*TG.*-UC)

gpo_index	physical pin on the camera
3	12 (OUT1)
1	11 (INOUT1)
4	4 (OUT2)

Models: MU cameras (MU.*-.*)

gpo_index	physical pin on the camera
1	1 (GX1)
2	3 (GX2)
3	5 (GX3)
4	7 (GX4)

Models: MX X2G2 cameras (MX.*X2G2.*)

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

Models: MX X4G2 cameras (MX.*X4G2.*)

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: MX X4G3 cameras (MX.*X4G3.*)

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)

Models: MX X8G3 cameras (MX.*X8G3-FF.*)

gpo_index	physical pin on the camera
1	IO-A 4 (OUT1 - optical)
2	IO-A 6 (OUT2 - optical)
3	IO-B 1 (INOUT1)
4	IO-B 3 (INOUT2)
5	IO-B 4 (INOUT3)
6	IO-B 6 (INOUT4)

Models: CB X8G3 cameras (CB.*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)

Models: CB X4G2 cameras (CB.*)

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: MX377 MTP cameras (MX377.*MTP.*)

gpo_index	physical pin on the camera
2	19 (OUT2 - optical)
3	17 (OUT3 - optical)
4	2 (INOUT1)
5	3 (INOUT2)
6	4 (INOUT3)
7	5 (INOUT4)
8	12 (INOUT5)
9	14 (INOUT6)
10	9 (INOUT7)
11	7 (INOUT8)

XI_PRM_GPO_MODE or "gpo_mode"¶
Description: Defines GPO functionality.

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 XI_PRM_TRG_SOURCE On models xiMU (MU9) xiQ.

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.

Type: Enumerator.
Default value: XI_GPO_OFF

Is invalidated by: XI_PRM_GPO_SELECTOR
Usage:

```
int gpo_mode = 0;
xiGetParamInt(handle, XI_PRM_GPO_MODE, &gpo_mode);
xiSetParamInt(handle, XI_PRM_GPO_MODE, XI_GPO_OFF);
```

Value	Description
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.
XI_GPO_FRAME_BUFFER_OVERFLOW	Frame buffer overflow status.
XI_GPO_EXPOSURE_ACTIVE_FIRST_ROW	Output is on while the first row exposure.
XI_GPO_EXPOSURE_ACTIVE_FIRST_ROW_NEG	Output is off while the first row exposure.
XI_GPO_EXPOSURE_ACTIVE_ALL_ROWS	Output is on while all rows exposure together.
XI_GPO_EXPOSURE_ACTIVE_ALL_ROWS_NEG	Output is off while all rows exposure together.
XI_GPO_TXD	Output is connected to TXD of UART module

Example:

```
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);
```

XI_PRM_LED_SELECTOR or "led_selector"¶
Description: Selects LED.

Type: Enumerator.
Default value: 1

Usage:

```
int led_selector = 0;
xiGetParamInt(handle, XI_PRM_LED_SELECTOR, &led_selector);
xiSetParamInt(handle, XI_PRM_LED_SELECTOR, XI_LED_SEL1);
```

Value	Description
XI_LED_SEL1	LED 1
XI_LED_SEL2	LED 2
XI_LED_SEL3	LED 3
XI_LED_SEL4	LED 4
XI_LED_SEL5	LED 5

XI_PRM_LED_MODE or "led_mode"¶
Description: Defines LED functionality.

Type: Enumerator.
Default value: XI_LED_HEARTBEAT

Is invalidated by: XI_PRM_LED_SELECTOR
Usage:

```
int led_mode = 0;
xiGetParamInt(handle, XI_PRM_LED_MODE, &led_mode);
xiSetParamInt(handle, XI_PRM_LED_MODE, XI_LED_HEARTBEAT);
```

Value	Description
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_LINK	Set led to blink if link is OK.
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_OFF	Set led to off.
XI_LED_ON	Set led to on.
XI_LED_BLINK	Blinking (1Hz).

XI_PRM_DEBOUNCE_EN or "dbnc_en"¶

Description: Enable/Disable debounce to selected GPI (XI_PRM_GPI_SELECTOR parameter). (see Note 1)

Note1: Parameter is available only for xiQ camera models.

Type: Integer.
Default value: XI_OFF

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_DEBOUNCE_EN, &value);
xiSetParamInt(handle, XI_PRM_DEBOUNCE_EN, XI_ON);
```

Debounce Setup¶

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.

Type: Integer.
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_DEBOUNCE_T0, &value);
xiSetParamInt(handle, XI_PRM_DEBOUNCE_T0, value);
```

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

Type: Integer.

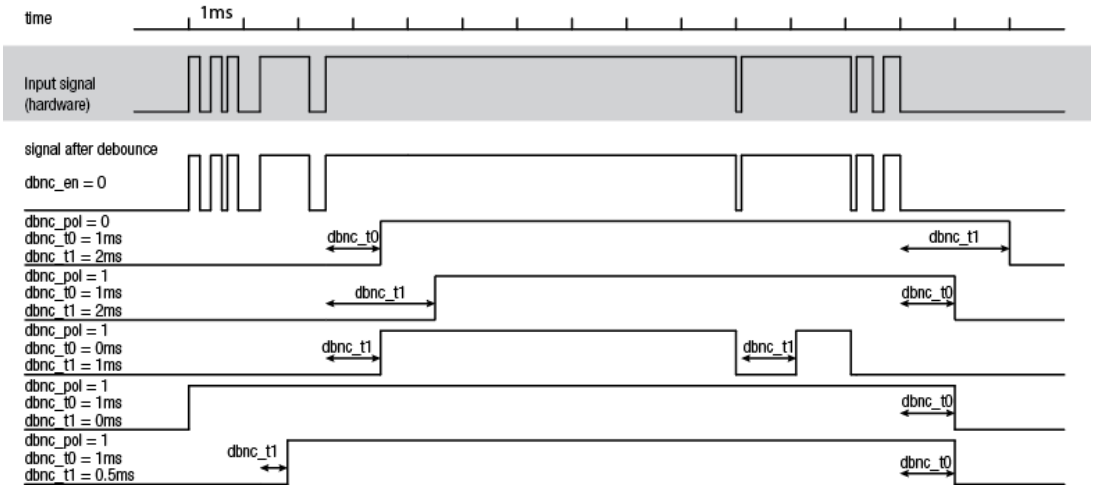
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_DEBOUNCE_T1, &value);
xiSetParamInt(handle, XI_PRM_DEBOUNCE_T1, value);
```

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 value: 0

Typical range: [0, 1]

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_DEBOUNCE_POL, &value);
xiSetParamInt(handle, XI_PRM_DEBOUNCE_POL, value);
```

Lens Control¶

Note: Some of XIMEA cameras can be equipped with controlled lens. API for lens control contains couple of parameters.

Lens tested OK with CB cameras:

- 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-S 24mm f/2.8 STM
- CANON EF-S 10-18mm f/4.5-5.6 IS STM
- CANON EF-S 18-135mm f/3.5-5.6 IS STM
- Canon EF 200mm f/2.8L II USM
- Canon EF 180mm f/3.5L Macro USM
- 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 value: XI_OFF

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_LENS_MODE, &value);
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 value: 1.0

Usage:

```
float value = 0.0;
```

```
xiGetParamFloat(handle, XI_PRM_LENS_APERTURE_VALUE, &value);
xiSetParamFloat(handle, XI_PRM_LENS_APERTURE_VALUE, value);
```

XI_PRM_LENS_APERTURE_INDEX or "lens_aperture_index"¶
Description: Current lens aperture motor step value.

Type: Integer.
Default value: 1

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_LENS_APERTURE_INDEX, &value);
xiSetParamInt(handle, XI_PRM_LENS_APERTURE_INDEX, value);
```

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 value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_LENS_FOCUS_MOVEMENT_VALUE, &value);
xiSetParamInt(handle, XI_PRM_LENS_FOCUS_MOVEMENT_VALUE, value);
```

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: Integer.
Default value: 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_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 value: 1.0

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: Enumerator.
Default value: XI_LENS_FEATURE_MOTORIZED_FOCUS_SWITCH

Usage:

```
See parameter XI_PRM_LENS_FEATURE
```

Value	Description
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	(planned feature) 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: Float.
Default value: 0.0

Usage:

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

Device info parameters¶

XI_PRM_DEVICE_NAME or "device_name"¶

Description: Return device name.

Type: String.
Default value: -

Usage:

```
char value[200] = "";
xiGetParamString(handle, XI_PRM_DEVICE_NAME, &value, sizeof(value));
```

XI_PRM_DEVICE_TYPE or "device_type"¶

Description: Returns device type (1394, USB2.0, USB3.0, PCIe, ...).

Type: String.
Default value: -

Usage:

```
char value[200] = "";
xiGetParamString(handle, XI_PRM_DEVICE_TYPE, &value, sizeof(value));
```

XI_PRM_DEVICE_MODEL_ID or "device_model_id"¶

Description: Returns the device model id.

Type: Integer.

Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_DEVICE_MODEL_ID, &value);
```

XI_PRM_SENSOR_MODEL_ID or "sensor_model_id"¶
Description: Returns the device sensor model id.

Type: Integer.
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_SENSOR_MODEL_ID, &value);
```

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 value: 0

Usage:

```
char sn[100] = "";
xiGetParamString(handle, XI_PRM_DEVICE_SN, sn, sizeof(sn));
```

Example:

```
// show serial number of this camera
char sn[100] = "";
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 value: 0

Usage:

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

XI_PRM_DEVICE_INSTANCE_PATH or "device_inst_path"¶

Description: Returns device instance path in operating system.

Type: String.
Default value: -

Usage:

```
char path[100] = "";
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 value: -

Usage:

```
char path[100] = "";
xiGetParamString(handle, XI_PRM_DEVICE_LOCATION_PATH, path,
sizeof(path));
```

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 a handle for opening or identification.

Note1: It is currently available only on some models

Note2: For xiQ camera devices are supported maximum length 56 characters.

Note3: For xiC camera devices are supported maximum length 48 characters.

Note4: For xiB, xiT, xiX camera devices are supported maximum length 4 characters. (Power off/on required after User ID changed)

Type: String.
Default value: -

Usage:

```
char name[100] = "";
xiGetParamString(handle,XI_PRM_DEVICE_USER_ID,name,sizeof(name));
```

Example: Example of setting and retrieving the DEVICE_USER_ID

Only after a successful reboot of the camera, it can be opened with the newly assigned

XI_PRM_DEVICE_MANIFEST or "device_manifest"¶
Description: Get XML of current xiAPI device parameters and capabilities.

Type: String.
Default value: -

Usage:

```
char* manifest_data = NULL;
#define MANIF_MAX_SIZE 2*1024*1024
manifest_data = malloc(MANIF_MAX_SIZE);
xiGetParamString(handle,XI_PRM_DEVICE_MANIFEST,manifest_data,MANIF_MA
X_SIZE);
```

XI_PRM_IMAGE_USER_DATA or "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.

Supported cameras: xiB, xiT

Type: Integer.
Default value: 0
Typical range: [0, 0xFFFFFFFF]

Usage:

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

Device acquisition settings¶

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 value: 0
Typical range: [0, 0xFFFF]

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_IMAGE_DATA_FORMAT_RGB32_ALPHA, &value);
xiSetParamInt(handle, XI_PRM_IMAGE_DATA_FORMAT_RGB32_ALPHA, value);
```

XI_PRM_IMAGE_PAYLOAD_SIZE or "imgpayloadsize"¶
Description: Buffer size in bytes sufficient for output image returned by GetImage

Type: Integer.
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_IMAGE_PAYLOAD_SIZE, &value);
```

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.
Default value: XI_GenTL_Image_Format_Mono8

Usage:

```
int transport_pixel_format = 0;
xiGetParamInt(handle, XI_PRM_TRANSPORT_PIXEL_FORMAT,
&transport_pixel_format);
xiSetParamInt(handle, XI_PRM_TRANSPORT_PIXEL_FORMAT,
XI_GenTL_Image_Format_Mono8);
```

XI_PRM_TRANSPORT_DATA_TARGET or "transport_data_target"¶
Description: Sets image data delivery target to CPU RAM (default) or GPU RAM.

How to configure GPUDirect for memory transfers How to configure CUDA for memory transfers
Type: Enumerator.
Default value: XI_TRANSPORT_DATA_TARGET_CPU_RAM

Usage:

```
int transport_data_target = 0;
xiGetParamInt(handle, XI_PRM_TRANSPORT_DATA_TARGET,
&transport_data_target);
xiSetParamInt(handle, XI_PRM_TRANSPORT_DATA_TARGET,
XI_TRANSPORT_DATA_TARGET_CPU_RAM);
```

Value	Description
XI_TRANSPORT_DATA_TARGET_CPU_RAM	normal CPU memory buffer is used for image data
XI_TRANSPORT_DATA_TARGET_GPU_RAM	data is delivered straight to GPU memory using GPUDirect technology
XI_TRANSPORT_DATA_TARGET_UNIFIED	CUDA managed memory is used for image data.
XI_TRANSPORT_DATA_TARGET_ZEROCOPY	CUDA zerocopy memory is used for image data.

XI_PRM_SENSOR_CLOCK_FREQ_HZ or "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 value: Depends on sensor model.

Is invalidated by: XI_PRM_LIMIT_BANDWIDTH
Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_SENSOR_CLOCK_FREQ_HZ, &value);
xiSetParamFloat(handle, XI_PRM_SENSOR_CLOCK_FREQ_HZ, value);
```

XI_PRM_SENSOR_CLOCK_FREQ_INDEX or "sensor_clock_freq_index"¶

Description: Sensor clock frequency. Selects frequency on cameras which supports only some specific frequencies.

Type: Integer.
Default value: Depends on sensor model.

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_SENSOR_CLOCK_FREQ_INDEX, &value);
xiSetParamInt(handle, XI_PRM_SENSOR_CLOCK_FREQ_INDEX, value);
```

XI_PRM_SENSOR_OUTPUT_CHANNEL_COUNT or "sensor_output_channel_count"¶
Description: Number of output channels from sensor used for data transfer.

Type: Enumerator.
Default value: Depends on sensor model.

Usage:

```
int sensor_output_channel_count = 0;
xiGetParamInt(handle, XI_PRM_SENSOR_OUTPUT_CHANNEL_COUNT,
&sensor_output_channel_count);
xiSetParamInt(handle, XI_PRM_SENSOR_OUTPUT_CHANNEL_COUNT,
XI_CHANN_CNT2);
```

Value	Description
XI_CHANN_CNT2	2 sensor readout channels.
XI_CHANN_CNT4	4 sensor readout channels.
XI_CHANN_CNT8	8 sensor readout channels.
XI_CHANN_CNT16	16 sensor readout channels.
XI_CHANN_CNT24	24 sensor readout channels.
XI_CHANN_CNT32	32 sensor readout channels.
XI_CHANN_CNT48	48 sensor readout channels.

XI_PRM_FRAMERATE or "framerate"¶
Description: Defines frames per second of sensor. See more details in article Frame Rate Control On some camera models it is possible to change or limit acquisition frame rate. Frame rate value should be within possible range, use XI_PRM_INFO_MAX, XI_PRM_INFO_MIN.

Note1: Use following code to set the frame rate to 10 FPS on MQ, MD cameras.

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

Note2: Use following code to limit the frame rate to 10 FPS on CB,MT,MX,MC cameras.

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

Type: Float.

Default value: 0.0

Is invalidated by: XI_PRM_IMAGE_DATA_FORMAT, XI_PRM_EXPOSURE, XI_PRM_ACQ_TIMING_MODE, XI_PRM_LIMIT_BANDWIDTH, XI_PRM_LIMIT_BANDWIDTH_MODE, XI_PRM_DOWNSAMPLING_TYPE, XI_PRM_DOWNSAMPLING, XI_PRM_BINNING_VERTICAL, XI_PRM_BINNING_HORIZONTAL, XI_PRM_DECIMATION_VERTICAL, XI_PRM_DECIMATION_HORIZONTAL, XI_PRM_WIDTH, XI_PRM_HEIGHT, XI_PRM_OUTPUT_DATA_PACKING, XI_PRM_OUTPUT_DATA_PACKING_TYPE, XI_PRM_SENSOR_DATA_BIT_DEPTH, XI_PRM_OUTPUT_DATA_BIT_DEPTH, XI_PRM_DUAL_ADC_MODE, XI_PRM_SENSOR_FEATURE_VALUE

Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_FRAMERATE, &value);
xiSetParamFloat(handle, XI_PRM_FRAMERATE, value);
```

XI_PRM_COUNTER_SELECTOR or "counter_selector"¶

Description: Selects which frame counter must be returned

Note1: It returns number of skipped frames on the transport layer, number of skipped frames on API layer, number of successfully transferred frames.

Type: Enumerator.

Default value: XI_CNT_SEL_TRANSPORT_SKIPPED_FRAMES

Usage:

```
int counter_selector = 0;
xiGetParamInt(handle, XI_PRM_COUNTER_SELECTOR, &counter_selector);
xiSetParamInt(handle, XI_PRM_COUNTER_SELECTOR,
XI_CNT_SEL_TRANSPORT_SKIPPED_FRAMES);
```

Value	Description
XI_CNT_SEL_TRANSPORT_SKIPPED_FRAMES	Number of skipped frames on transport layer (e.g. when image gets lost while transmission). Occur when capacity of transport channel does not allow to transfer all data.
XI_CNT_SEL_API_SKIPPED_FRAMES	Number of skipped frames on API layer. Occur when application does not process the images as quick as they are received from the camera.
XI_CNT_SEL_TRANSPORT_TRANSFERRED_FRAMES	Number of delivered buffers since last acquisition start.

XI_CNT_SEL_FRAME_MISSED_TRIGGER_DUETO_OVERLAP	Number of missed triggers overlapped with exposure or read-out stage of previous frame – see XI_PRM_TRG_OVERLAP. (see Note1)
XI_CNT_SEL_FRAME_MISSED_TRIGGER_DUETO_FRAME_BUFFER_OVR	Number of missed triggers due to frame buffer full. (see Note1)
XI_CNT_SEL_FRAME_BUFFER_OVERFLOW	Internal camera frame buffer memory (RAM) full events counter. It can be incremented multiple times per one frame. (see Note1)
XI_CNT_SEL_TRANSPORT_QUEUE_UNDERRUN	Incremented when camera starts to transfer new image, however no target buffer is queued in the transport queue. Connected to GenTL.STREAM_INFO_NUM_UNDERRUN. (see Note1)
XI_CNT_SEL_ACQUISITION_AUTO_RESTARTED_ON_FAILURE	Acquisition can be restarted, due to failures on bus

Note1: Available only on cameras series: xiX, xiB, xiT, xiC, xiMUdeveloped since 2023 (MU196, MU050, MU051).

XI_PRM_COUNTER_VALUE or "counter_value"¶
Description: Returns value of selected (by XI_PRM_COUNTER_SELECTOR) frame counter.

Note: All counters are reset with the camera open, and counters XI_CNT_SEL_TRANSPORT_SKIPPED_FRAMES, XI_CNT_SEL_API_SKIPPED_FRAMES and XI_CNT_SEL_TRANSPORT_TRANSFERRED_FRAMES are also reset with acquisition start.

Type: Integer.
Default value: 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);
```

XI_PRM_ACQ_TIMING_MODE or "acq_timing_mode"¶
Description: This parameter defines the acquisition timing mode. More information about enumerators XI_ACQ_TIMING_MODE_FRAME_RATE and XI_ACQ_TIMING_MODE_FRAME_RATE_LIMIT please refer to our Frame Rate Control support page.

Type: Enumerator.
Default value: XI_ACQ_TIMING_MODE_FREE_RUN

Usage:

```
int acq_timing_mode = 0;
xiGetParamInt(handle, XI_PRM_ACQ_TIMING_MODE, &acq_timing_mode);
xiSetParamInt(handle, XI_PRM_ACQ_TIMING_MODE,
XI_ACQ_TIMING_MODE_FREE_RUN);
```

Value	Description
XI_ACQ_TIMING_MODE_FREE_RUN	camera acquires images at a maximum possible framerate
XI_ACQ_TIMING_MODE_FRAME_RATE	Selects a mode when sensor frame acquisition frequency is set to parameter FRAMERATE
XI_ACQ_TIMING_MODE_FRAME_RATE_LIMIT	Selects a mode when sensor frame acquisition frequency is limited by parameter FRAMERATE

XI_PRM_AVAILABLE_BANDWIDTH or "available_bandwidth"¶
Description: Measure available interface bandwidth. Unit is Megabits (1000000) per sec.

Note: Some parameters could be changed by getting available bandwidth. Please set camera parameters to needed value after getting of available bandwidth.

Type: Integer.
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_AVAILABLE_BANDWIDTH, &value);
```

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. Size of the image buffer can be obtained by parameter XI_PRM_IMAGE_PAYLOAD_SIZE

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

Type: Enumerator.
Default value: XI_BP_UNSAFE

Usage:

```
int buffer_policy = 0;
xiGetParamInt(handle, XI_PRM_BUFFER_POLICY, &buffer_policy);
```

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

Value	Description
XI_BP_UNSAFE	User gets pointer to internally allocated circle buffer and data may be overwritten by device.
XI_BP_SAFE	Data from device will be copied to user allocated buffer or xiApi allocated memory.

XI_PRM_LUT_EN or "LUTEnable"¶

Description: Activates Look-Up-Table (LUT).

Note1: Possible value: 0 - sensor pixels are transferred directly

Note2: Possible value: 1 - sensor pixels are mapped through LUT

Note3: LUT parameters are valid only for some cameras. E.g. xiQ supports LUT. xiMU (MU9PM-MH) does NOT support it.

Note4: For xiQ cameras setting XI_PRM_LUT_EN also uploads previously set values in to camera. Values are latched in API.

Type: Integer.
Default value: XI_OFF

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_LUT_EN, &value);
xiSetParamInt(handle, XI_PRM_LUT_EN, XI_ON);
```

XI_PRM_LUT_INDEX or "LUTIndex"¶

Description: Controls the index (offset) of the coefficient to access in the LUT.

Note1: All xiQ cameras have LUT N-bit to N-bit, based on the XI_PRM_SENSOR_DATA_BIT_DEPTH For the specific camera. All xiC/xiX/xiT cameras have LUT 12-bit to 12-bit.

Note2: Range of applicable indexes depends on sensor digitization bit depth (sensor_bit_depth). Use XI_PRM_INFO_MAX, XI_PRM_INFO_MIN.

Type: Integer.
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_LUT_INDEX, &value);
```

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

XI_PRM_LUT_VALUE or "LUTValue"¶
Description: Defines value at entry LUTIndex of the LUT.

Note1: Range of applicable values depends on sensor digitization bit depth (sensor_bit_depth). Use XI_PRM_INFO_MAX, XI_PRM_INFO_MIN.

Note2: All xiQ cameras have LUT N-bit to N-bit, based on the XI_PRM_SENSOR_DATA_BIT_DEPTH of the specific camera. All xiC/xiX/xiT cameras have LUT 12-bit to 12-bit.

Note2: For xiQ cameras setting values has no direct effect on image, only after setting XI_PRM_LUT_EN to value 1, will apply all changes to camera.

Type: Integer.
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_LUT_VALUE, &value);
xiSetParamInt(handle, XI_PRM_LUT_VALUE, value);
```

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.

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.

Type: Integer.
Default value: 0

Is invalidated by: 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
```

XI_PRM_TS_RST_MODE or "ts_rst_mode"¶
Description: Defines way timestamp reset engine is armed.

Type: Enumerator.

Default value: XI_TS_RST_ARM_ONCE

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);
```

Value	Description
XI_TS_RST_ARM_ONCE	Engine is disabled after TimeStamp has been reset after selected event.
XI_TS_RST_ARM_PERSIST	Engine is armed permanently so each selected event will trigger TimeStamp reset.

XI_PRM_TS_RST_SOURCE or "ts_rst_source"¶

Description: Defines source for timestamp reset engine as well as the polarity active signal. The engine is edge sensitive.

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

Type: Enumerator.

Default value: XI_TS_RST_OFF

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);
```

Value	Description
XI_TS_RST_OFF	No source selected TimeStamp 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	TimeStamp reset source selected GPO1
XI_TS_RST_SRC_GPO_2	TimeStamp reset source selected GPO2
XI_TS_RST_SRC_GPO_3	TimeStamp reset source selected GPO3
XI_TS_RST_SRC_GPO_4	TimeStamp reset source selected GPO4
XI_TS_RST_SRC_GPO_1_INV	TimeStamp reset source selected GPO1 inverted
XI_TS_RST_SRC_GPO_2_INV	TimeStamp reset source selected GPO2 inverted
XI_TS_RST_SRC_GPO_3_INV	TimeStamp reset source selected GPO3 inverted
XI_TS_RST_SRC_GPO_4_INV	TimeStamp reset source selected GPO4 inverted
XI_TS_RST_SRC_TRIGGER	TRIGGER to sensor rising edge is active
XI_TS_RST_SRC_TRIGGER_INV	TRIGGER to sensor rising edge is active
XI_TS_RST_SRC_SW	TRIGGER to sensor rising edge is active. TimeStamp 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)
XI_TS_RST_SRC_GPI_5	GPI5 rising edge is active
XI_TS_RST_SRC_GPI_6	GPI6 rising edge is active
XI_TS_RST_SRC_GPI_5_INV	GPI5 falling edge is active
XI_TS_RST_SRC_GPI_6_INV	GPI6 falling edge is active
XI_TS_RST_SRC_GPI_7	TimeStamp reset source selected GPI7 (after de bounce)
XI_TS_RST_SRC_GPI_8	TimeStamp reset source selected GPI8 (after de bounce)
XI_TS_RST_SRC_GPI_9	TimeStamp reset source selected GPI9 (after de bounce)
XI_TS_RST_SRC_GPI_10	TimeStamp reset source selected GPI10 (after de bounce)
XI_TS_RST_SRC_GPI_11	TimeStamp reset source selected GPI11 (after de bounce)

XI_TS_RST_SRC_GPI_7_INV	TimeStamp reset source selected GPI7 inverted (after de bounce)
XI_TS_RST_SRC_GPI_8_INV	TimeStamp reset source selected GPI8 inverted (after de bounce)
XI_TS_RST_SRC_GPI_9_INV	TimeStamp reset source selected GPI9 inverted (after de bounce)
XI_TS_RST_SRC_GPI_10_INV	TimeStamp reset source selected GPI10 inverted (after de bounce)
XI_TS_RST_SRC_GPI_11_INV	TimeStamp reset source selected GPI11 inverted (after de bounce)

Extended Device parameters¶

XI_PRM_IS_DEVICE_EXIST or "isexist"¶

Description: Returns 1 if camera connected and works properly.

Type: Integer.
Default value: XI_OFF

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_IS_DEVICE_EXIST, &value);
```

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.

Note1: 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.

Note2: The maximal value for this parameter is 2147483647 because it uses the signed integer.

Type: Integer.
Default value: 100000000
Typical range: [0, 2147483647]

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_ACQ_BUFFER_SIZE, &value);
```

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

Example: 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))

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

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 value: 1
Typical range: [1, 2147483647]

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 (only valid for MQ,MD camera families). 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.

Note: Whole range minimum to maximum is not guaranteed on all tested configurations. Please be aware of possible issues on some controllers.

Type: Integer.
Default value: 0

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. (only valid for MQ,MD camera families)

Type: Integer.
Default value: 0

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 value: 4
Typical range: [2, 2147483647]

Is invalidated by: XI_PRM_ACQ_BUFFER_SIZE
Usage:

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

Example: Application sets queue size to 7

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).

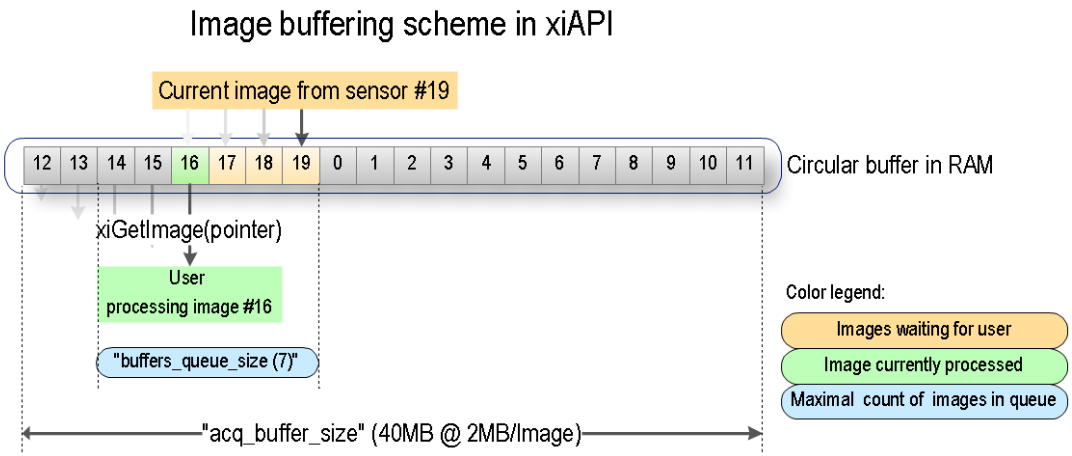


Illustration of normal status when application is processing one frame longer

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.

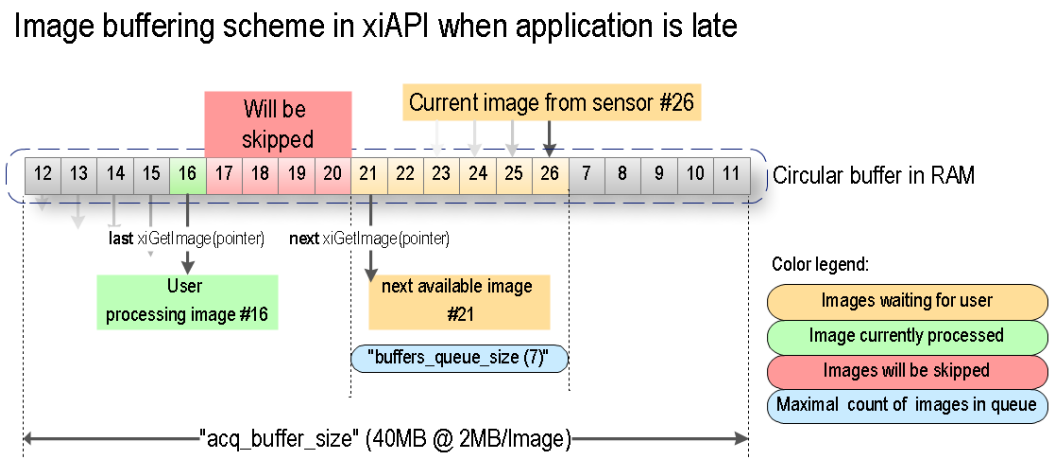


Illustration of state when application gets late. Longer than queue.

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 be committed to transport layer. (only valid for USB 3.0 camera families)

Type: Integer.

Default value: 1
Typical range: [1, 256]

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_ACQ_TRANSPORT_BUFFER_COMMIT, &value);
xiSetParamInt(handle, XI_PRM_ACQ_TRANSPORT_BUFFER_COMMIT, value);
```

XI_PRM_RECENT_FRAME or "recent_frame"¶
Description: This parameter changes the behavior of xiGetImage.

Note1: possible value: 0 - Retrieves next available image from buffer

Note2: possible value: 1 - Retrieves the most recent image from buffer

Type: Integer.
Default value: XI_OFF

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_RECENT_FRAME, &value);
xiSetParamInt(handle, XI_PRM_RECENT_FRAME, XI_ON);
```

XI_PRM_DEVICE_RESET or "device_reset"¶
Description: Resets the camera firmware. From the functional view, it is the same as disconnection and connection of the camera.It is typically followed by an enumeration of the 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. It should be used on the device with the stopped acquisition. After re-enumeration closing of the old device to release its handler is recommended. A new handler should be used for further image acquisition.

Note: currently supported only for xiQ camera family

Type: Integer.
Default value: 0

Usage:

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

```
xiCloseDevice(old_handle);
HANDLE new_handle = NULL;
xiOpenDevice(0, &new_handle);
```

XI_PRM_CONCAT_IMG_MODE or "concat_img_mode"¶

Description: Enable/disable the Concatenated Images in One Buffer feature

Type: Integer.
Default value: XI_OFF

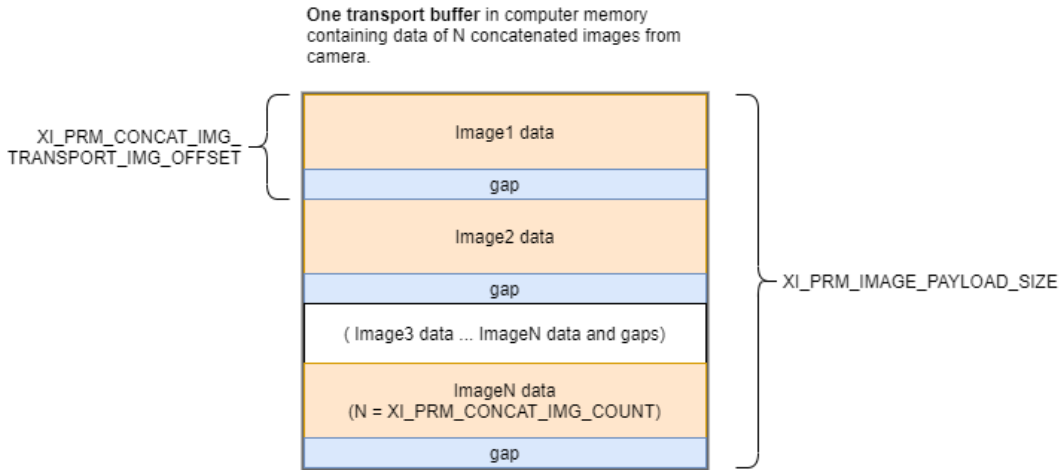
Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_CONCAT_IMG_MODE, &value);
xiSetParamInt(handle, XI_PRM_CONCAT_IMG_MODE, XI_ON);
```

XI_PRM_CONCAT_IMG_COUNT or "concat_img_count"¶

Description: Number of Concatenated Images in One Buffer.

Note: Read more at Concatenated Images in One Buffer feature.



Type: Integer.
Default value: 1
Typical range: [1, 0]

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_CONCAT_IMG_COUNT, &value);
xiSetParamInt(handle, XI_PRM_CONCAT_IMG_COUNT, value);
```

XI_PRM_CONCAT_IMG_TRANSPORT_IMG_OFFSET or "concat_img_transport_img_offset"¶

Description: Offset between images data in transport buffer when feature Concatenated

Images in One Buffer is enabled

Type: Integer.
Default value: 1

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_CONCAT_IMG_TRANSPORT_IMG_OFFSET,
&value);
```

XI_PRM_PROBE_SELECTOR or "probe_selector"¶
Description: Select Probe

Type: Enumerator.
Default value: XI_PROBE_SELECTOR_CURRENT_MAINBOARD_VCC_IN

Usage:

```
int probe_selector = 0;
xiGetParamInt(handle, XI_PRM_PROBE_SELECTOR, &probe_selector);
xiSetParamInt(handle, XI_PRM_PROBE_SELECTOR,
XI_PROBE_SELECTOR_CURRENT_MAINBOARD_VCC_IN);
```

Value	Description
XI_PROBE_SELECTOR_CURRENT_MAINBOARD_VCC_IN	Current probe on Main Board at VCC_IN power supply
XI_PROBE_SELECTOR_VOLTAGE_MAINBOARD_VCC_IN	Voltage probe on Main Board at VCC_IN power supply
XI_PROBE_SELECTOR_CURRENT_MAINBOARD_VCC_ADJ2	Current probe on Main Board at VCC_ADJ2 power supply
XI_PROBE_SELECTOR_VOLTAGE_MAINBOARD_VCC_ADJ2	Voltage probe on Main Board at VCC_ADJ2 power supply
XI_PROBE_SELECTOR_CURRENT_MAINBOARD_VCC_ADJ1	Current probe on Main Board at VCC_ADJ1 power supply
XI_PROBE_SELECTOR_VOLTAGE_MAINBOARD_VCC_ADJ1	Voltage probe on Main Board at VCC_ADJ1 power supply
XI_PROBE_SELECTOR_CURRENT_MAINBOARD_VCC_PLT	Current probe on Main Board at VCC_PLT power supply

XI_PROBE_SELECTOR_VOLTAGE_MAINBOARD_VCC_PLT	Voltage probe on Main Board at VCC_PLT power supply
XI_PROBE_SELECTOR_VOLTAGE_SENSORBOARD_VCC_ADJ1	Voltage probe on Sensor Board at VCC_ADJ1
XI_PROBE_SELECTOR_VOLTAGE_SENSORBOARD_VCC_ADJ2	Voltage probe on Sensor Board at VCC_ADJ2
XI_PROBE_SELECTOR_VOLTAGE_SENSORBOARD_VCC_5V0	Voltage probe on Sensor Board at VCC_5V0
XI_PROBE_SELECTOR_VOLTAGE_SENSORBOARD_VCC_3V3	Voltage probe on Sensor Board at VCC_3V3
XI_PROBE_SELECTOR_VOLTAGE_DATA_CON_INPUT	Voltage probe on device input, if device is bus powered
XI_PROBE_SELECTOR_VOLTAGE_PELTIER1	Voltage probe on peltier #1
XI_PROBE_SELECTOR_CURRENT_PELTIER1	Current probe on peltier #1
XI_PROBE_SELECTOR_VOLTAGE_PELTIER2	Voltage probe on peltier #2
XI_PROBE_SELECTOR_CURRENT_PELTIER2	Current probe on peltier #2

XI_PRM_PROBE_VALUE or "probe_value"¶

Description: Returns Value of the selected Probe

Type: Float.

Default value: 0.0

Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_PROBE_VALUE, &value);
```

Sensor Defects Correction¶

XI_PRM_COLUMN_FPN_CORRECTION or "column_fpn_correction"¶

Description: Correction of column fpn.

Type: Enumerator.

Default value: XI_OFF

Usage:

```
int column_fpn_correction = 0;
xiGetParamInt(handle, XI_PRM_COLUMN_FPN_CORRECTION,
```

```
&column_fpn_correction);
xiSetParamInt(handle, XI_PRM_COLUMN_FPN_CORRECTION, XI_OFF);
```

Value	Description
XI_OFF	Turn parameter off
XI_ON	Turn parameter on

XI_PRM_ROW_FPN_CORRECTION or "row_fpn_correction"¶
Description: Correction of row fpn.

Type: Enumerator.
Default value: XI_OFF

Usage:

```
int row_fpn_correction = 0;
xiGetParamInt(handle, XI_PRM_ROW_FPN_CORRECTION,
&row_fpn_correction);
xiSetParamInt(handle, XI_PRM_ROW_FPN_CORRECTION, XI_OFF);
```

Value	Description
XI_OFF	Turn parameter off
XI_ON	Turn parameter on

XI_PRM_COLUMN_BLACK_OFFSET_CORRECTION or "column_black_offset_correction"¶
Description: Correction of column black offset.

Type: Enumerator.
Default value: XI_OFF

Usage:

```
int column_black_offset_correction = 0;
xiGetParamInt(handle, XI_PRM_COLUMN_BLACK_OFFSET_CORRECTION,
&column_black_offset_correction);
xiSetParamInt(handle, XI_PRM_COLUMN_BLACK_OFFSET_CORRECTION, XI_OFF);
```

Value	Description
XI_OFF	Turn parameter off
XI_ON	Turn parameter on

XI_PRM_ROW_BLACK_OFFSET_CORRECTION or "row_black_offset_correction"¶

Description: Correction of row black offset.

Type: Enumerator.
Default value: XI_OFF

Usage:

```
int row_black_offset_correction = 0;
xiGetParamInt(handle, XI_PRM_ROW_BLACK_OFFSET_CORRECTION,
&row_black_offset_correction);
xiSetParamInt(handle, XI_PRM_ROW_BLACK_OFFSET_CORRECTION, XI_OFF);
```

Value	Description
XI_OFF	Turn parameter off
XI_ON	Turn parameter on

Sensor features¶

XI_PRM_SENSOR_MODE or "sensor_mode"¶

Description: Current sensor mode. Allows to select sensor mode by one integer. Setting of this parameter affects: image dimensions and downsampling.

Type: Enumerator.
Default value: 0

Usage:

```
int sensor_mode = 0;
xiGetParamInt(handle, XI_PRM_SENSOR_MODE, &sensor_mode);
xiSetParamInt(handle, XI_PRM_SENSOR_MODE, XI_SENS_MD0);
```

Value	Description
XI_SENS_MD0	Sensor mode number 0
XI_SENS_MD1	Sensor mode number 1
XI_SENS_MD2	Sensor mode number 2
XI_SENS_MD3	Sensor mode number 3
XI_SENS_MD4	Sensor mode number 4

XI_SENS_MD5	Sensor mode number 5
XI_SENS_MD6	Sensor mode number 6
XI_SENS_MD7	Sensor mode number 7
XI_SENS_MD8	Sensor mode number 8
XI_SENS_MD9	Sensor mode number 9
XI_SENS_MD10	Sensor mode number 10
XI_SENS_MD11	Sensor mode number 11
XI_SENS_MD12	Sensor mode number 12
XI_SENS_MD13	Sensor mode number 13
XI_SENS_MD14	Sensor mode number 14
XI_SENS_MD15	Sensor mode number 15

XI_PRM_HDR or "hdr"¶
Description: Enable High Dynamic Range sensor feature.

Note1: enables HDR mode for certain type of sensors. For more information see HDR mode support page.

Type: Integer.
Default value: XI_OFF

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_HDR, &value);
xiSetParamInt(handle, XI_PRM_HDR, XI_ON);
```

XI_PRM_HDR_KNEEPOINT_COUNT or "hdr_kneepoint_count"¶
Description: number of kneepoints.

Note1: Defines the number of kneepoints in the Piecewise Linear Response (PWLR) curve.

Note2: In case of one kneepoint, the kneepoint is defined by parameters (T2,SL2). In case of two kneepoints define both (T1,SL1), (T2,SL2).

Type: Integer.
Default value: 1
Typical range: [1, 2]

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_HDR_KNEEPOINT_COUNT, &value);
xiSetParamInt(handle, XI_PRM_HDR_KNEEPOINT_COUNT, value);
```

XI_PRM_HDR_T1 or "hdr_t1"¶

Description: Exposure time (T1) of 1st kneepoint in % of XI_PRM_EXPOSURE

Type: Integer.
Default value: 60
Typical range: [0, 100]

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_HDR_T1, &value);
xiSetParamInt(handle, XI_PRM_HDR_T1, value);
```

XI_PRM_HDR_T2 or "hdr_t2"¶

Description: Exposure time (T2) of 2nd kneepoint in % of

Type: Integer.
Default value: 80
Typical range: [0, 100]

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_HDR_T2, &value);
xiSetParamInt(handle, XI_PRM_HDR_T2, value);
```

XI_PRM_KNEEPOINT1 or "hdr_kneepoint1"¶

Description: Saturation level (SL1) of 1st kneepoint in % of sensor saturation.

Type: Integer.
Default value: 40
Typical range: [0, 100]

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_KNEEPOINT1, &value);
xiSetParamInt(handle, XI_PRM_KNEEPOINT1, value);
```

XI_PRM_KNEEPOINT2 or "hdr_kneepoint2"¶

Description: Saturation level (SL2) of 2nd kneepoint in % of sensor saturation.

Type: Integer.
Default value: 60
Typical range: [0, 100]

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_KNEEPOINT2, &value);
xiSetParamInt(handle, XI_PRM_KNEEPOINT2, value);
```

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. Setting of this parameter does not affect the data from sensor or API when camera is connected. It can be used for setting black level only for Offline Processing.

Type: Integer.
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_IMAGE_BLACK_LEVEL, &value);
```

XI_PRM_IMAGE_AREA or "image_area"¶

Description: Defines image area of sensor as output.

Type: Enumerator.
Default value: XI_IMAGE_AREA_ACTIVE

Usage:

```
int image_area = 0;
xiGetParamInt(handle, XI_PRM_IMAGE_AREA, &image_area);
xiSetParamInt(handle, XI_PRM_IMAGE_AREA, XI_IMAGE_AREA_ACTIVE);
```

Value	Description
XI_IMAGE_AREA_ACTIVE	All light sensitive pixels suggested by image vendor.
XI_IMAGE_AREA_ACTIVE_AND_MASKED	All Active pixels plus masked pixels surrounding the Active area.

XI_PRM_DUAL_ADC_MODE or "dual_adc_mode"¶

Description: Sets DualADC Mode

Type: Enumerator.
Default value: XI_DUAL_ADC_MODE_OFF

Usage:

```
int dual_adc_mode = 0;
xiGetParamInt(handle, XI_PRM_DUAL_ADC_MODE, &dual_adc_mode);
xiSetParamInt(handle, XI_PRM_DUAL_ADC_MODE, XI_DUAL_ADC_MODE_OFF);
```

Value	Description
XI_DUAL_ADC_MODE_OFF	Disable DualADC feature
XI_DUAL_ADC_MODE_COMBINED	Set Combined mode
XI_DUAL_ADC_MODE_NON_COMBINED	Set NonCombined mode

XI_PRM_DUAL_ADC_GAIN_RATIO or "dual_adc_gain_ratio"¶

Description: Sets DualADC Gain Ratio in dB

Type: Float.
Default value: 0.0
Typical range: [0.0, 24.0]

Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_DUAL_ADC_GAIN_RATIO, &value);
xiSetParamFloat(handle, XI_PRM_DUAL_ADC_GAIN_RATIO, value);
```

XI_PRM_DUAL_ADC_THRESHOLD or "dual_adc_threshold"¶

Description: Sets DualADC Threshold value

Type: Integer.
Default value: 50
Typical range: [0, 100]

Is invalidated by: XI_PRM_DUAL_ADC_MODE
Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_DUAL_ADC_THRESHOLD, &value);
xiSetParamInt(handle, XI_PRM_DUAL_ADC_THRESHOLD, value);
```

XI_PRM_COMPRESSION_REGION_SELECTOR or "compression_region_selector"¶

Description: Sets Compression Region Selector

Type: Integer.
Default value: 1
Typical range: [1, 2]

Is invalidated by: XI_PRM_DUAL_ADC_MODE
Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_COMPRESSION_REGION_SELECTOR, &value);
xiSetParamInt(handle, XI_PRM_COMPRESSION_REGION_SELECTOR, value);
```

XI_PRM_COMPRESSION_REGION_START or "compression_region_start"¶

Description: Sets Compression Region Start

Type: Float.
Default value: 0.0
Typical range: [0.0, 50.0]

Is invalidated by: XI_PRM_DUAL_ADC_MODE, XI_PRM_DUAL_ADC_GAIN_RATIO, XI_PRM_COMPRESSION_REGION_SELECTOR
Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_COMPRESSION_REGION_START, &value);
xiSetParamFloat(handle, XI_PRM_COMPRESSION_REGION_START, value);
```

XI_PRM_COMPRESSION_REGION_GAIN or "compression_region_gain"¶

Description: Sets Compression Region Gain

Type: Float.
Default value: 0.0
Typical range: [-90.0, 0.0]

Is invalidated by: XI_PRM_DUAL_ADC_MODE
Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_COMPRESSION_REGION_GAIN, &value);
xiSetParamFloat(handle, XI_PRM_COMPRESSION_REGION_GAIN, value);
```

Version info¶

XI_PRM_VERSION_SELECTOR or "version_selector"¶

Description: Selects module/unit, which version we get.

Type: Enumerator.
Default value: 0

Usage:

```
int version_selector = 0;
xiGetParamInt(handle, XI_PRM_VERSION_SELECTOR, &version_selector);
xiSetParamInt(handle, XI_PRM_VERSION_SELECTOR, XI_VER_API);
```

Value	Description
XI_VER_API	version of API
XI_VER_DRV	version of device driver
XI_VER_MCU1	version of MCU1 firmware.
XI_VER_MCU2	version of MCU2 firmware.
XI_VER_MCU3	version of MCU3 firmware.
XI_VER_FPGA1	version of FPGA1 firmware.
XI_VER_XMLMAN	version of XML manifest.
XI_VER_HW_REV	version of hardware revision.
XI_VER_FACTORY_SET	version of factory set.

XI_PRM_VERSION or "version"¶

Description: Returns version of selected module/unit(XI_PRM_VERSION_SELECTOR).

Type: String.
Default value: -

Usage:

```
char value[200] = "";
xiGetParamString(handle, XI_PRM_VERSION, &value, sizeof(value));
```

XI_PRM_API_VERSION or "api_version"¶

Description: Returns the version of API.

Type: String.
Default value: -

Usage:

```
char value[200] = "";
xiGetParamString(handle, XI_PRM_API_VERSION, &value, sizeof(value));
```

XI_PRM_DRV_VERSION or "drv_version"¶
Description: Returns the version of the current device driver.

Type: String.
Default value: -

Usage:

```
char value[200] = "";
xiGetParamString(handle, XI_PRM_DRV_VERSION, &value, sizeof(value));
```

XI_PRM_MCU1_VERSION or "version_mcu1"¶
Description: Returns the version of the current MCU1 firmware.

Type: String.
Default value: -

Usage:

```
char value[200] = "";
xiGetParamString(handle, XI_PRM_MCU1_VERSION, &value, sizeof(value));
```

XI_PRM_MCU2_VERSION or "version_mcu2"¶
Description: Returns the version of the current MCU2 firmware.

Type: String.
Default value: -

Usage:

```
char value[200] = "";
xiGetParamString(handle, XI_PRM_MCU2_VERSION, &value, sizeof(value));
```

XI_PRM_MCU3_VERSION or "version_mcu3"¶
Description: Returns the version of the current MCU3 firmware.

Type: String.

Default value: -

Usage:

```
char value[200] = "";
xiGetParamString(handle, XI_PRM_MCU3_VERSION, &value, sizeof(value));
```

XI_PRM_FPGA1_VERSION or "version_fpga1"¶
Description: Returns version of FPGA firmware currently running.

Type: String.
Default value: -

Usage:

```
char value[200] = "";
xiGetParamString(handle, XI_PRM_FPGA1_VERSION, &value,
sizeof(value));
```

XI_PRM_XMLMAN_VERSION or "version_xmlman"¶
Description: Returns version of XML manifest.

Type: String.
Default value: -

Usage:

```
char value[200] = "";
xiGetParamString(handle, XI_PRM_XMLMAN_VERSION, &value,
sizeof(value));
```

XI_PRM_HW_REVISION or "hw_revision"¶
Description: Returns the hardware revision number of the camera.

Type: String.
Default value: -

Usage:

```
char value[200] = "";
xiGetParamString(handle, XI_PRM_HW_REVISION, &value, sizeof(value));
```

XI_PRM_FACTORY_SET_VERSION or "factory_set_version"¶
Description: Returns version of factory set.

Type: String.
Default value: -

Usage:

```
char value[200] = "";
xiGetParamString(handle, XI_PRM_FACTORY_SET_VERSION, &value,
sizeof(value));
```

API features¶

XI_PRM_DEBUG_LEVEL or "debug_level"¶

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

Type: Enumerator.
Default value: XI_DL_WARNING

Usage:

```
int debug_level = 0;
xiGetParamInt(handle, XI_PRM_DEBUG_LEVEL, &debug_level);
xiSetParamInt(handle, XI_PRM_DEBUG_LEVEL, XI_DL_DETAIL);
```

Value	Description
XI_DL_DETAIL	(see Note1)
XI_DL_TRACE	Prints errors, warnings and important informations
XI_DL_WARNING	Prints all errors and warnings
XI_DL_ERROR	Prints all errors
XI_DL_FATAL	Prints only important errors
XI_DL_DISABLED	Prints no messages

Note1: Prints same as XI_DL_TRACE plus locking of resources.

Note2: In Windows use DebugView to view the current messages.

Note3: In Linux the messages are printed to stderr

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 measurement and calculation before opening the

camera (xiOpenDevice), resulting in reducing the time to open a camera. Setting to XI_ON the measurement is enabled (default).

Note1: 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.

Note2: When set to value XI_OFF, the time required to open a camera (xiOpenDevice) is reduced.

Type: Integer.
Default value: XI_ON

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_AUTO_BANDWIDTH_CALCULATION, &value);
xiSetParamInt(handle, XI_PRM_AUTO_BANDWIDTH_CALCULATION, XI_ON);
```

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 value: XI_ON

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_NEW_PROCESS_CHAIN_ENABLE, &value);
xiSetParamInt(handle, XI_PRM_NEW_PROCESS_CHAIN_ENABLE, XI_ON);
```

XI_PRM_PROC_NUM_THREADS or "proc_num_threads"¶
Description: Number of threads per image processor. An application can change this number in order to optimize performance or decrease number of threads to save resources.

Note: this parameter does not work for MQ, MD camera families and for MU9Px-MH camera.

Type: Integer.
Default value: 0

Typical range: [1, 61]

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_PROC_NUM_THREADS, &value);
xiSetParamInt(handle, XI_PRM_PROC_NUM_THREADS, value);
```

Camera FFS¶

Note: Some of XIMEA cameras contain Flash File System. It allows to store/read small customer file in each camera. For more information visit our knowledge base article How to work with FFS.

XI_PRM_READ_FILE_FFS or "read_file_ffs"¶
Description: File data to be read from camera flash file system.

Type: String.
Default value: -

Usage:

```
char data_buff[100] = "";
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);
```

Example: 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.
Default value: -

Usage:

```
char value[200] = "";
xiGetParamString(handle, XI_PRM_WRITE_FILE_FFS, &value,
sizeof(value));
xiSetParamString(handle, XI_PRM_WRITE_FILE_FFS, value,
strlen(value));
```

Example: Write file:

```
xiSetParamString(xiH, XI_PRM_FFS_FILE_NAME, "User1",
strlen("User1"));
char* file_content = "ABCDEFGH";
xiSetParamString(xiH, XI_PRM_WRITE_FILE_FFS, file_content,
strlen(file_content));
```

Example: 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_NAME or "ffs_file_name"¶
Description: Name of file to be written/read from camera FFS.

Note: On MX,MC,CB,MT cameras family, there is limited set of filenames. User's application

can use filenames: User1, User2, User3 to store some application specific data.

Type: String.
Default value: -

Usage:

```
char filename[100] = "User1";
xiSetParamString(handle, XI_PRM_FFS_FILE_NAME, filename, size);
```

XI_PRM_FFS_FILE_ID or "ffs_file_id"¶
Description: File number(id) in camera FFS.

Type: Integer.
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_FFS_FILE_ID, &value);
```

Example: 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: Size of a file specified with parameter XI_PRM_FFS_FILE_IDin bytes.

Type: Integer.
Default value: 0

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_FFS_FILE_SIZE, &value);
```

XI_PRM_FREE_FFS_SIZE or "free_ffs_size"¶

Description: Size of free camera flash file system space in bytes.

Type: Unsigned integer 64 bit.
Default value: 0

Usage:

```
uint64_t value = 0;
DWORD size = sizeof(value);
XI_PRM_TYPE type = xiTypeInteger64;
xiGetParam(handle, XI_PRM_FREE_FFS_SIZE, &value, &size, &type);
```

XI_PRM_USED_FFS_SIZE or "used_ffs_size"¶

Description: Size of used camera flash file system space in bytes.

Type: Unsigned integer 64 bit.
Default value: 0

Usage:

```
uint64_t value = 0;
DWORD size = sizeof(value);
XI_PRM_TYPE type = xiTypeInteger64;
xiGetParam(handle, XI_PRM_USED_FFS_SIZE, &value, &size, &type);
```

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 value: 0

Usage:

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

APIContextControl¶

XI_PRM_API_CONTEXT_LIST or "xiapi_context_list"¶

Description: API Context contains the text representation of current settings for offline

image processing. It can be gotten while acquisition to store the context. Respectively, it can be set while offline processing - to restore the context.

Type: String.
Default value:

Usage:

```
char value[200] = "";
xiGetParamString(handle, XI_PRM_API_CONTEXT_LIST, &value,
sizeof(value));
xiSetParamString(handle, XI_PRM_API_CONTEXT_LIST, value,
strlen(value));
```

Sensor Control¶

Note: Some of XIMEA cameras have 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. See more at our support page, SENSOR FEATURE SELECTOR.

Type: Enumerator.
Default value: XI_SENSOR_FEATURE_ZEROROT_ENABLE

Usage:

```
int sensor_feature_selector = 0;
xiGetParamInt(handle, XI_PRM_SENSOR_FEATURE_SELECTOR,
&sensor_feature_selector);
xiSetParamInt(handle, XI_PRM_SENSOR_FEATURE_SELECTOR,
XI_SENSOR_FEATURE_ZEROROT_ENABLE);
```

Value	Description
XI_SENSOR_FEATURE_ZEROROT_ENABLE	Sensor Zero ROT enable for ONSEMI PYTHON family. For camera model:MQ013xG-ON (on/off)
XI_SENSOR_FEATURE_BLACK_LEVEL_CLAMP	Black level offset clamping (value). for Camera model:MD

XI_SENSOR_FEATURE_MD_FPGA_DIGITAL_GAIN_DISABLE	Disable digital component of gain for MD family (1=disabled/0=enabled)
XI_SENSOR_FEATURE_ACQUISITION_RUNNING	Sensor acquisition is running status (0/1). Could be stopped by setting of 0. For camera model:CB,MC,MX,MT
XI_SENSOR_FEATURE_TIMING_MODE	Sensor timing mode (value depends on sensor)
XI_SENSOR_FEATURE_PARALLEL_ADC	Enables the parallel ADC readout mode, where all exposed pixels undergo dual sampling, leading to reduced readout noise at the cost of increased readout time
XI_SENSOR_FEATURE_BLACK_LEVEL_OFFSET_RAW	Sensor specific register raw black level offset (value)
XI_SENSOR_FEATURE_SHORT_INTERVAL_SHUTTER	Sensor short Interval Shutter (on/off)
XI_SENSOR_FEATURE_AUTO_LOW_POWER_MODE_AUTO	Sensor low power mode (on/off)
XI_SENSOR_FEATURE_HIGH_CONVERSION_GAIN	Enables high conversion gain feature which applies additional gain to the signal at the pixel level. This leads to a reduction in read noise and a boost in sensitivity and signal-to-noise ratio, particularly in low-light situations. Consequently, the camera exhibits superior performance in dark environments, capturing images with minimal noise and enhanced detail.
XI_SENSOR_FEATURE_DUAL_TRG_EXP_ZONE_DIVIDER_POSITION	Sensor Dual Trigger Exposure Zone Divider Position
XI_SENSOR_FEATURE_TOF_VCSEL_CTRL_VOLTAGE_MV	ToF VCSEL Control Voltage in mV

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 value: 0

Typical range: [0, 1024]

Is invalidated by: XI_PRM_SENSOR_FEATURE_SELECTOR

Usage:

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

Extended Features¶

XI_PRM_ACQUISITION_STATUS_SELECTOR or "acquisition_status_selector"¶

Description: Selects the internal acquisition signal to read using XI_PRM_ACQUISITION_STATUS

Type: Enumerator.

Default value: XI_ACQUISITION_STATUS_ACQ_ACTIVE

Usage:

```
int acquisition_status_selector = 0;
xiGetParamInt(handle, XI_PRM_ACQUISITION_STATUS_SELECTOR,
&acquisition_status_selector);
xiSetParamInt(handle, XI_PRM_ACQUISITION_STATUS_SELECTOR,
XI_ACQUISITION_STATUS_ACQ_ACTIVE);
```

Value	Description
XI_ACQUISITION_STATUS_ACQ_ACTIVE	Device is currently doing an acquisition of one or many frames.

XI_PRM_ACQUISITION_STATUS or "acquisition_status"¶

Description: Returns status of acquisition.

Type: Enumerator.

Default value: XI_OFF

Usage:

```
int value = 0;
xiGetParam(handle, XI_PRM_ACQUISITION_STATUS, &value, sizeof(int),
xiTypeInteger);
xiGetParamInt(handle, XI_PRM_ACQUISITION_STATUS, &value);
```

Value	Description

XI_OFF	Turn parameter off
XI_ON	Turn parameter on

XI_PRM_DP_UNIT_SELECTOR or "dp_unit_selector"¶

Description: Data Pipe Unit Selector.

Type: Enumerator.
Default value: XI_DP_UNIT_SENSOR

Usage:

```
int dp_unit_selector = 0;
xiGetParamInt(handle, XI_PRM_DP_UNIT_SELECTOR, &dp_unit_selector);
xiSetParamInt(handle, XI_PRM_DP_UNIT_SELECTOR, XI_DP_UNIT_SENSOR);
```

Value	Description
XI_DP_UNIT_SENSOR	Selects device image sensor
XI_DP_UNIT_FPGA	Selects device image FPGA

XI_PRM_DP_PROC_SELECTOR or "dp_proc_selector"¶

Description: Data Pipe Processor Selector.

Type: Enumerator.
Default value: XI_DP_PROC_NONE

Is invalidated by: XI_PRM_DP_UNIT_SELECTOR
Usage:

```
int dp_proc_selector = 0;
xiGetParamInt(handle, XI_PRM_DP_PROC_SELECTOR, &dp_proc_selector);
xiSetParamInt(handle, XI_PRM_DP_PROC_SELECTOR, XI_DP_PROC_NONE);
```

Value	Description
XI_DP_PROC_NONE	Default empty processor
XI_DP_PROC_CHANNEL_MUXER	Channel Muxer (selected processor combines multiple input channels)
XI_DP_PROC_PIXEL_SEQUENCER	Selects pixel data output sequence
XI_DP_PROC_CHANNEL_1	Selects sensor output channel 1
XI_DP_PROC_CHANNEL_2	Selects sensor output channel 2

XI_DP_PROC_FRAME_BUFFER	Selects frame buffer memory

XI_PRM_DP_PARAM_SELECTOR or "dp_param_selector"¶
Description: Data Pipe Processor parameter Selector.

Type: Enumerator.
Default value: XI_DP_PARAM_NONE

Is invalidated by: XI_PRM_DP_UNIT_SELECTOR, XI_PRM_DP_PROC_SELECTOR
Usage:

```
int dp_param_selector = 0;
xiGetParamInt(handle, XI_PRM_DP_PARAM_SELECTOR, &dp_param_selector);
xiSetParamInt(handle, XI_PRM_DP_PARAM_SELECTOR, XI_DP_PARAM_NONE);
```

Value	Description
XI_DP_PARAM_NONE	Empty parameter
XI_DP_PARAM_CHMUX_CHANNEL_SELECTOR	Defines output of Channel Muxer processor
XI_DP_PARAM_CHMUX_ALPHA	Channel merger coefficient Alpha
XI_DP_PARAM_CHMUX_BETA	Channel merger coefficient Beta
XI_DP_PARAM_PIXSEQ_SELECTOR	PixSeq Selector
XI_DP_PARAM_CHANNEL_TIMING	Selected channel timing
XI_DP_PARAM_FRAMEBUF_MODE	Frame Buffer Mode
XI_DP_PARAM_FRAMEBUF_SIZE	Frame Buffer Size Bytes

XI_PRM_DP_PARAM_VALUE or "dp_param_value"¶
Description: Data Pipe processor parameter value.

Type: Float.
Default value: 0.0
Typical range: [0.0, 100000.0]

Is invalidated by: XI_PRM_DP_UNIT_SELECTOR, XI_PRM_DP_PROC_SELECTOR, XI_PRM_DP_PARAM_SELECTOR
Usage:

```
float value = 0.0;
xiGetParamFloat(handle, XI_PRM_DP_PARAM_VALUE, &value);
xiSetParamFloat(handle, XI_PRM_DP_PARAM_VALUE, value);
```

XI_PRM_GENTL_DATASTREAM_ENABLED or "gentl_stream_en"¶

Description: Control of GenTL data stream. Enabling by XI_ON the acquisition buffering must be controlled by GenTL interface (e.g. DSAllocAndAnnounceBuffer, DSQueueBuffer)

Type: Integer.
Default value: XI_OFF

Usage:

```
int value = 0;
xiGetParamInt(handle, XI_PRM_GENTL_DATASTREAM_ENABLED, &value);
xiSetParamInt(handle, XI_PRM_GENTL_DATASTREAM_ENABLED, XI_ON);
```

XI_PRM_GENTL_DATASTREAM_CONTEXT or "gentl_stream_context"¶

Description: Pointer to GenTL stream context. It can be used later with GenTL buffers handling.

Note: See more details in the example xiAPI-capture-50-images-gentl.

Type: String.
Default value: 0

Usage:

```
xiSetParamInt(handle, XI_PRM_GENTL_DATASTREAM_ENABLED, XI_ON);
void* stream_h = NULL;
xiGetParamString(handle, XI_PRM_GENTL_DATASTREAM_CONTEXT, &stream_h,
sizeof(void*)));
void* buffer_handle = NULL;
AllocAndAnnounceBuffer(stream_h, payload_size, NULL, &buffer_handle);
```

User Set Control¶

Note: Parameters for for global control of the device settings. They allow loading or saving factory or user-defined settings to the camera memory.

XI_PRM_USER_SET_SELECTOR or "user_set_selector"¶

Description: User Set to be loaded by XI_PRM_USER_SET_LOAD.

Note: Available only on some camera models: MX377, MJ042, MJ150.

Type: Enumerator.
Default value: 0

Usage:

```
int user_set_selector = 0;
xiGetParamInt(handle, XI_PRM_USER_SET_SELECTOR, &user_set_selector);
xiSetParamInt(handle, XI_PRM_USER_SET_SELECTOR, XI_US_12_STD_L);
```

Value	Description
XI_US_12_STD_L	12bit per channel STD Low Gain mode preset.
XI_US_12_STD_H	12bit per channel STD High Gain mode preset.
XI_US_14_STD_L	14bit per channel STD Low Gain mode preset.
XI_US_NONE	No preset selected.
XI_US_14_STD_H	14bit per channel STD High Gain mode preset.
XI_US_2_12_CMS_S_L	12bit per channel, 2 samples, CMS(summing) Low Gain mode preset.
XI_US_2_12_CMS_S_H	12bit per channel, 2 samples, CMS(summing) High Gain mode preset.
XI_US_2_14_CMS_S_L	14bit per channel, 2 samples, CMS(summing) Low Gain mode preset.
XI_US_2_14_CMS_S_H	14bit per channel, 2 samples, CMS(summing) High Gain mode preset.
XI_US_4_12_CMS_S_L	12bit per channel, 4 samples, CMS(summing) Low Gain mode preset.
XI_US_4_12_CMS_S_H	12bit per channel, 4 samples, CMS(summing) High Gain mode preset.
XI_US_4_14_CMS_S_L	14bit per channel, 4 samples, CMS(summing) Low Gain mode preset.
XI_US_4_14_CMS_S_H	14bit per channel, 4 samples, CMS(summing) High Gain mode preset.
XI_US_2_12_HDR_HL	12bit per channel, 2 samples, HDR High Low Gain mode preset.
XI_US_2_12_HDR_L	12bit per channel, 2 samples, HDR Low Gain mode preset.
XI_US_2_12_HDR_H	12bit per channel, 2 samples, HDR High Gain mode preset.
XI_US_4_12_CMS_HDR_HL	12bit per channel, 4 samples, CMS + HDR High Low Gain mode preset.
XI_US_2_14_HDR_L	14bit per channel, 2 samples, HDR Low Gain mode preset.
XI_US_2_14_HDR_H	14bit per channel, 2 samples, HDR High Gain mode preset.

XI_US_2_12_CMS_A_L	12bit per channel, 2 samples, CMS(averaging) Low Gain mode preset.
XI_US_2_12_CMS_A_H	12bit per channel, 2 samples, CMS(averaging) High Gain mode preset.

XI_PRM_USER_SET_LOAD or "user_set_load"¶

Description: Loads User Set selected by XI_PRM_USER_SET_SELECTOR. User Set is list of API parameters and values, which is applied similarily as xiSetParam one by one. If setting of some parameter fails, the process of loading is aborted and error value is returned to the application. All parameters changed remains without any restore to previous state.

Note: Available only on some camera models: MX377, MJ042, MJ150.

Type: Integer.
Default value: 0

Usage:

```
int value = 0;
xiSetParamInt(handle, XI_PRM_USER_SET_LOAD, value);
```

XI_PRM_USER_SET_DEFAULT or "user_set_default"¶

Description: Selected User Set to load and make active when the device is opened. Change might affect default mode in other applications, e.g. CamTool.

Note: Available only on some camera models: MX377, MJ042, MJ150.

Type: Enumerator.
Default value: 0

Usage:

```
int user_set_default = 0;
xiGetParamInt(handle, XI_PRM_USER_SET_DEFAULT, &user_set_default);
xiSetParamInt(handle, XI_PRM_USER_SET_DEFAULT, XI_US_12_STD_L);
```

Value	Description
XI_US_12_STD_L	12bit per channel STD Low Gain mode preset.
XI_US_12_STD_H	12bit per channel STD High Gain mode preset.
XI_US_14_STD_L	14bit per channel STD Low Gain mode preset.
XI_US_NONE	No preset selected.

XI_US_14_STD_H	14bit per channel STD High Gain mode preset.
XI_US_2_12_CMS_S_L	12bit per channel, 2 samples, CMS(summing) Low Gain mode preset.
XI_US_2_12_CMS_S_H	12bit per channel, 2 samples, CMS(summing) High Gain mode preset.
XI_US_2_14_CMS_S_L	14bit per channel, 2 samples, CMS(summing) Low Gain mode preset.
XI_US_2_14_CMS_S_H	14bit per channel, 2 samples, CMS(summing) High Gain mode preset.
XI_US_4_12_CMS_S_L	12bit per channel, 4 samples, CMS(summing) Low Gain mode preset.
XI_US_4_12_CMS_S_H	12bit per channel, 4 samples, CMS(summing) High Gain mode preset.
XI_US_4_14_CMS_S_L	14bit per channel, 4 samples, CMS(summing) Low Gain mode preset.
XI_US_4_14_CMS_S_H	14bit per channel, 4 samples, CMS(summing) High Gain mode preset.
XI_US_2_12_HDR_HL	12bit per channel, 2 samples, HDR High Low Gain mode preset.
XI_US_2_12_HDR_L	12bit per channel, 2 samples, HDR Low Gain mode preset.
XI_US_2_12_HDR_H	12bit per channel, 2 samples, HDR High Gain mode preset.
XI_US_4_12_CMS_HDR_HL	12bit per channel, 4 samples, CMS + HDR High Low Gain mode preset.
XI_US_2_14_HDR_L	14bit per channel, 2 samples, HDR Low Gain mode preset.
XI_US_2_14_HDR_H	14bit per channel, 2 samples, HDR High Gain mode preset.
XI_US_2_12_CMS_A_L	12bit per channel, 2 samples, CMS(averaging) Low Gain mode preset.
XI_US_2_12_CMS_A_H	12bit per channel, 2 samples, CMS(averaging) High Gain mode preset.

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_SETTABLE¶

Description: Check if parameter is settable. It finishes with success when settable.

Usage:

```
if (XI_OK == xiSetParamInt(handle, XI_PRM_TEMP_SELECTOR
XI_PRM_INFO_SETTABLE, XI_TEMP_SENSOR_BOARD))
{
        printf("Camera supports TEMP_SENSOR_BOARD\n");
}
```

XI_PRM_INFO_MIN¶

Description: Acquire parameter minimum value

Usage:

```
int exp_min = 0;
xiGetParamInt(handle, XI_PRM_EXPOSURE XI_PRM_INFO_MIN, &exp_min);
float framerate = 0;
xiGetParamFloat(handle, XI_PRM_FRAMERATE XI_PRM_INFO_MIN,
&framerate);
```

XI_PRM_INFO_MAX¶

Description: Acquire parameter maximum value.

Usage:

```
int exp_max = 0;
xiGetParamInt(handle, XI_PRM_EXPOSURE XI_PRM_INFO_MAX, &exp_max);
float framerate = 0;
xiGetParamFloat(handle, XI_PRM_FRAMERATE XI_PRM_INFO_MAX,
&framerate);
```

XI_PRM_INFO_INCREMENT¶

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

Usage:

```
int height_inc = 0;
xiGetParamInt(handle, XI_PRM_HEIGHT XI_PRM_INFO_INCREMENT,
&height_inc);
```

XI_PRM_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:

```
int exp_val = 0;
xiSetParamInt(handle, XI_PRM_EXPOSURE XI_PRMM_DIRECT_UPDATE,
exp_val);
int gain_val = 0;
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) see Note
    - Gain (XI_PRM_GAIN) see 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

Generated:f712c97: Mon Apr 15 03:05:02 CEST 2024