Helpdesk XIMEA

xiAPI.NET Manual

Support SK - 2026-01-26 - in xiAPI.NET

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

xiAPI.NET Manual¶
Table of Contents¶

- string GetParamString(string prm)
- void GetParam(string prm, out byte[] val, out int val_len)
- void GetParam(string prm, out string val, out int val_len)
- ImgParams GetLastImageParams()

ImgParams class

- int GetDataFormat()
- int GetWidth()
- int GetHeight()
- int GetFrameNum()
- double GetTimestamp()
- int GetDigitalInputLevel()
- int GetBlackLevel()

xiCamEnum class

- int ReEnumerate()
- string GetSerialNumById(int devIndex)
- string GetDevNameById(int devIndex)
- string GetInstancePathById(int devIndex)
- string GetDeviceTypeById(int devIndex)
- string GetDeviceUserIdById(int devIndex)

xiAPI.NET Parameters Basic

- EXPOSURE
- EXPOSURE_TIME_SELECTOR
- EXPOSURE_BURST_COUNT
- GAIN_SELECTOR
- GAIN
- DOWNSAMPLING
- DOWNSAMPLING_TYPE
- TEST_PATTERN_GENERATOR_SELECTOR
- TEST_PATTERN
- IMAGE_DATA_FORMAT
- IMAGE_DATA_SIGN
- SHUTTER_TYPE
- SENSOR_TAPS
- AEAG
- AEAG_ROI_OFFSET_X
- AEAG_ROI_OFFSET_Y
- AEAG_ROI_WIDTH
- AEAG_ROI_HEIGHT

- SENS_DEFECTS_CORR_LIST_SELECTOR
- SENS_DEFECTS_CORR_LIST_CONTENT
- SENS_DEFECTS_CORR
- AUTO_WB
- MANUAL_WB
- WB_ROI_OFFSET_X
- WB_ROI_OFFSET_Y
- WB_ROI_WIDTH
- WB_ROI_HEIGHT
- WB_KR
- WB_KG
- WB_KB
- WIDTH
- HEIGHT
- OFFSET_X
- OFFSET_Y
- REGION_SELECTOR
- REGION_MODE
- REGIONS_GEOMETRY
- HORIZONTAL_FLIP
- VERTICAL_FLIP
- INTERLINE_EXPOSURE_MODE
- FFC
- FFC_FLAT_FIELD_FILE_NAME
- FFC_DARK_FIELD_FILE_NAME
- TOF_READOUT_MODE
- TOF_MODULATION_FREQUENCY
- TOF_MULTIPLE_PHASES_IN_BUFFER
- TOF_PHASES_COUNT
- TOF_PHASE_ANGLE
- TOF_PHASE_EXPOSURE_TIME
- TOF_PHASE_SELECTOR

Image Format

- BINNING_SELECTOR
- BINNING_VERTICAL_MODE
- BINNING_VERTICAL
- BINNING_VERTICAL_FLOAT
- BINNING_HORIZONTAL_MODE
- BINNING_HORIZONTAL
- BINNING_HORIZONTAL_FLOAT

- BINNING_HORIZONTAL_PATTERN
- BINNING_VERTICAL_PATTERN
- DECIMATION_SELECTOR
- DECIMATION_VERTICAL
- DECIMATION_HORIZONTAL
- DECIMATION_HORIZONTAL_PATTERN
- DECIMATION_VERTICAL_PATTERN

AE Setup

- EXP_PRIORITY
- AG_MAX_LIMIT
- AE_MAX_LIMIT
- AEAG_LEVEL
- AEAG_SKIP_FRAMES_COUNT

Performance

- LIMIT_BANDWIDTH
- LIMIT_BANDWIDTH_MODE
- SENSOR_DATA_BIT_DEPTH
- OUTPUT_DATA_BIT_DEPTH
- IMAGE_DATA_BIT_DEPTH
- OUTPUT_DATA_PACKING
- OUTPUT_DATA_PACKING_TYPE

Temperature

- IS_COOLED
- COOLING
- TARGET_TEMP
- TEMP_SELECTOR
- TEMP
- TEMP_CONTROL_MODE
- CHIP_TEMP
- HOUS_TEMP
- HOUS_BACK_SIDE_TEMP
- SENSOR_BOARD_TEMP
- TEMP_ELEMENT_SEL
- TEMP_ELEMENT_VALUE

Color Correction

- CMS
- CMS_INTENT
- APPLY_CMS

- INPUT_CMS_PROFILE
- OUTPUT_CMS_PROFILE
- IMAGE_IS_COLOR
- COLOR_FILTER_ARRAY
- GAMMAY
- GAMMAC
- SHARPNESS
- CC_MATRIX_00
- DEFAULT_CC_MATRIX
- CC_MATRIX_NORM

Device IO

- TRG_SOURCE
- TRG_SOFTWARE
- TRG_SELECTOR
- TRG_OVERLAP
- ACQ_FRAME_BURST_COUNT
- TIMESTAMP

GPIO Setup

- GPI_SELECTOR
- GPI_MODE
- GPI_LEVEL
- GPI_LEVEL_AT_IMAGE_EXP_START
- GPI_LEVEL_AT_IMAGE_EXP_END
- GPI_LEVEL_AT_IMAGE_READOUT_END
- GPO_SELECTOR
- GPO_MODE
- METADATA_SAMPLING_MODE
- LED_SELECTOR
- LED_MODE
- DEBOUNCE_EN

Debounce Setup

- DEBOUNCE_T0
- DEBOUNCE_T1
- DEBOUNCE_POL

Lens Control

- LENS_MODE
- LENS_APERTURE_VALUE
- LENS_APERTURE_INDEX

- LENS_FOCUS_MOVEMENT_VALUE
- LENS_FOCUS_MOVE
- LENS_FOCAL_LENGTH
- LENS_FEATURE_SELECTOR
- LENS_FEATURE

Device info parameters

- DEVICE_NAME
- DEVICE_TYPE
- DEVICE_MODEL_ID
- SENSOR_MODEL_ID
- DEVICE_SN
- DEVICE_SENS_SN
- DEVICE_INSTANCE_PATH
- DEVICE_LOCATION_PATH
- DEVICE_USER_ID
- DEVICE_MANIFEST
- IMAGE_USER_DATA

Device acquisition settings

- IMAGE_DATA_FORMAT_RGB32_ALPHA
- IMAGE_PAYLOAD_SIZE
- TRANSPORT_PIXEL_FORMAT
- TRANSPORT_DATA_TARGET
- SENSOR_CLOCK_FREQ_HZ
- SENSOR_CLOCK_FREQ_INDEX
- SENSOR_OUTPUT_CHANNEL_COUNT
- FRAMERATE
- COUNTER_SELECTOR
- COUNTER_VALUE
- ACQ_TIMING_MODE
- AVAILABLE_BANDWIDTH
- BUFFER_POLICY
- LUT_EN
- LUT_INDEX
- LUT_VALUE
- TRG_DELAY
- TS_RST_MODE
- TS_RST_SOURCE

Extended Device parameters

- IS_DEVICE_EXIST
- ACQ_BUFFER_SIZE
- ACQ_BUFFER_SIZE_UNIT
- ACQ_TRANSPORT_BUFFER_SIZE
- ACQ_TRANSPORT_PACKET_SIZE
- BUFFERS_QUEUE_SIZE
- ACQ_TRANSPORT_BUFFER_COMMIT
- RECENT_FRAME
- DEVICE_RESET
- CONCAT_IMG_MODE
- CONCAT_IMG_COUNT
- CONCAT_IMG_TRANSPORT_IMG_OFFSET
- PROBE_SELECTOR
- PROBE_VALUE

Sensor Defects Correction

- COLUMN_FPN_CORRECTION
- ROW_FPN_CORRECTION
- COLUMN_BLACK_OFFSET_CORRECTION
- ROW_BLACK_OFFSET_CORRECTION

Sensor features

- SENSOR_MODE
- HDR
- HDR_KNEEPOINT_COUNT
- HDR_T1
- HDR_T2
- KNEEPOINT1
- KNEEPOINT2
- IMAGE_BLACK_LEVEL
- IMAGE_AREA
- DUAL_ADC_MODE
- DUAL_ADC_GAIN_RATIO
- DUAL_ADC_THRESHOLD
- COMPRESSION_REGION_SELECTOR
- COMPRESSION_REGION_START
- COMPRESSION_REGION_GAIN

Version info

- VERSION_SELECTOR
- VERSION

- API_VERSION
- DRV_VERSION
- MCU1_VERSION
- MCU2_VERSION
- MCU3_VERSION
- FPGA1_VERSION
- XMLMAN_VERSION
- HW_REVISION
- FACTORY_SET_VERSION

API features

- DEBUG_LEVEL
- AUTO_BANDWIDTH_CALCULATION
- NEW_PROCESS_CHAIN_ENABLE
- PROC_NUM_THREADS

Camera FFS

- READ_FILE_FFS
- WRITE_FILE_FFS
- FFS_FILE_NAME
- FFS_FILE_ID
- FFS_FILE_SIZE
- FREE_FFS_SIZE
- USED_FFS_SIZE
- FFS_ACCESS_KEY

APIContextControl

- API_CONTEXT_LIST

Sensor Control

- SENSOR_FEATURE_SELECTOR
- SENSOR_FEATURE_VALUE

Extended Features

- ACQUISITION_STATUS_SELECTOR
- ACQUISITION_STATUS
- DP_UNIT_SELECTOR
- DP_PROC_SELECTOR
- DP_PARAM_SELECTOR
- DP_PARAM_VALUE
- GENTL_DATASTREAM_ENABLED
- GENTL_DATASTREAM_CONTEXT

User Set Control

- USER_SET_SELECTOR
- USER_SET_LOAD
- USER_SET_DEFAULT

API parameter modifiers

- SETTABLE
- MIN
- MAX
- INCREMENT
- REQ_VAL_BUFFER_SIZE
- DIRECT_UPDATE

Writing Applications with xiAPI.NET¶

Default parameters¶

After camera is opened by xiCam.OpenDevice 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.

---

xiCam class¶

Contains function definitions for the xiAPI.NET class.

void GetNumberDevices(out int devCount)¶

Description: Returns the number of all discovered devices.

Parameters:

out int devCount: The number of connected devices.

int GetNumberDevices()¶

Description: Returns the number of all discovered devices.

Return: The number of connected devices.

void OpenDevice(int DevID)¶

Description: This function initializes the device.

Parameters:

int DevID: Index of the device.

void OpenDeviceBy(OPEN_BY OPEN_BY_INST_PATH, String Param)¶

Description: This function initializes the device.

Parameters:

OpenDevBy Open_By: Method to be used when selecting the device to be opened, use OpenDevBy enumerator for selection.

String Param: Input parameter for device selection, e.g. serial number or instance path.

Corresponding Enumerator OPEN_BY

Value	Description
OPEN_BY_INST_PATH	Open camera by its hardware path
OPEN_BY_SN	Open camera by its serial number
OPEN_BY_USER_ID	open camera by its custom user ID
OPEN_BY_LOC_PATH	Open camera by its hardware location path

Note1: When the value of the parameter is changed and we want to open the camera with function OpenDevice with the value , 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 void GetNumberDevices(out int devCount) function each time you call additional OpenDevice.

void CloseDevice()¶
Description: This function will uninitialize the specified device and release allocated resources.

void StartAcquisition()¶
Description: This function starts the data acquisition on the device.

void StopAcquisition()¶
Description: Ends the work cycle of the camera, stops data acquisition and deallocates internal image buffers.

void GetImage(out BitmapSource image, int timeout)¶
Description: This function acquires an image and fills BitmapSource object.

Parameters:
out BitmapSource image: WPF BitmapSource to be filled.
int timeout: Time interval required to wait for the image (in milliseconds).

void GetImage(out WriteableBitmap image, int timeout)¶
Description: This function acquires an image and fills WritableBitmap object. Supports UNSAFE buffer policy mode.

Parameters:
out WriteableBitmap image: WPF BitmapSource to be filled.
int timeout: Time interval required to wait for the image (in milliseconds).

void GetImage(WriteableBitmap image, int timeout)¶
Description: This function acquires an image and fills WritableBitmap object. Supports

SAFE buffer policy mode.

Parameters:
WriteableBitmap image: WPF BitmapSource to be filled.
int timeout: Time interval required to wait for the image (in milliseconds).

void GetImage(out Bitmap image, int timeout)¶
Description: This function acquires an image and fills bitmap object. Supports UNSAFE buffer policy mode.

Parameters:
out Bitmap image: GDI+ bitmap to be filled
int timeout: Time interval required to wait for the image (in milliseconds)

void GetImage(Bitmap image, int timeout)¶
Description: This function acquires an image and fills bitmap object. Supports SAFE buffer policy mode.

Parameters:
Bitmap image: GDI+ bitmap to be filled.
int timeout: Time interval required to wait for the image (in milliseconds)

void GetImage(out byte[] image, int timeout)¶
Description: This function acquires an image into a byte array. Supports UNSAFE buffer policy mode.

Parameters:
out byte[] image: Preallocated byte array to be filled.
int timeout: Time interval required to wait for the image (in milliseconds)

void GetImage(byte[] image, int timeout)¶
Description: This function acquires an image into a byte array. Supports SAFE buffer policy mode.

Parameters:
byte[] image: Preallocated byte array to be filled.
int timeout: Time interval required to wait for the image (in milliseconds)

void GetXI_IMG(ref XI_IMG image, int timeout)¶
Description: This function acquires an image and fills XI_IMG structure.

Parameters:
ref XI_IMG image: XI_IMG structure to be filled. Needs to be cleared before use (*image.Clear()*)
int timeout: Time interval required to wait for the image (in milliseconds)

Note: Allocation of buffers is influenced by buffering policy. See details of behavior at parameter PRM.BUFFER_POLICY.

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.
	sequence_type	Type of image sequence (e.g. seq_disabled, seq_multiple_exposures)
4	sequence_image_id	Current image in sequence. Starting from 1. Zero if sequence is disabled.
4	sequence_length	Count of images in sequence. Zero for sequence disabled.

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

Note2: If the buffer policy is set to 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 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 IMAGE_USER_DATA parameter.

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

XI_IMG GetXI_IMG(int timeout)¶
Description: This function acquires an image and returns XI_IMG structure.

Parameters:
int timeout: Time interval required to wait for the image (in milliseconds).

Return: XI_IMG structure with information about acquired image (see description above).

void SetParam(string prm, int val)¶
Description: This function configures the device.

Parameters:
string prm: Parameter name string. Use class PRM to set parameters.
int val: Integer value to be set.

void SetParam(string prm, float val)¶
Description: This function configures the device.

Parameters:
string prm: Parameter name string. Use class PRM to set parameters.
float val: Float value to be set.

void SetParam(string prm, string val)¶
Description: This function configures the device.

Parameters:
string prm: Parameter name string. Use class PRM to set parameters.
string val: string value to be set.

void GetParam(string prm, out int val)¶
Description: This function returns parameter value as integer (current value, minimum, maximum, info).

Parameters:
string prm: Parameter name string. Use class PRM to set parameters.
out int val: Integer value to be returned.

int GetParamInt(string prm)¶
Description: This function returns parameter value as integer (current value, minimum, maximum, info).

Parameters:
string prm: Parameter name string. Use class PRM to set parameters.

Return: Integer value to be returned.

void GetParam(string prm, out ulong val)¶

Description: This function returns parameter value as unsigned long (current value, minimum, maximum, info).

Parameters:
string prm: Parameter name string. Use class PRM to set parameters.
out ulong val: Unsigned long value to be returned.

ulong GetParamUlong(string prm)¶

Description: This function returns parameter value as unsigned long (current value, minimum, maximum, info).

Parameters:
string prm: Parameter name string. Use class PRM to set parameters.

Return: unsigned long value to be returned.

void GetParam(string prm, out float val)¶

Description: This function returns parameter value as float (current value, minimum, maximum, info).

Parameters:
string prm: Parameter name string. Use class PRM to set parameters.
out float val: Float value to be returned.

float GetParamFloat(string prm)¶

Description: This function returns parameter value as float (current value, minimum, maximum, info).

Parameters:
string prm: Parameter name string. Use class PRM to set parameters.

Return: Float value to be returned.

void GetParam(string prm, out string val)¶

Description: This function returns parameter value as string (current value, minimum, maximum, info).

Parameters:
string prm: Parameter name string. Use class PRM to set parameters.
out string val: String value to be returned.

string GetParamString(string prm)¶

Description: This function returns parameter value as string.

Parameters:
string prm: Parameter name string. Use class PRM to set parameters.

Return: String value to be returned.

void GetParam(string prm, out byte[] val, out int val_len)¶

Description: This function returns parameter value as byte array.

Parameters:

string prm: Parameter name string. Use class PRM to set parameters.

out byte[] val: Byte array value to be returned.

out int val_len: Length of returned byte array value.

void GetParam(string prm, out string val, out int val_len)¶

Description: This function returns parameter value as string.

Parameters:

string prm: DOPLN Parameter name string. Use class PRM to set parameters.

out string val: String value to be returned.

out int val_len: Length of returned string value.

ImgParams GetLastImageParams()¶

Description: This function returns more detailed information about the last image in the ImgParams object.

Return: ImgParams object.

ImgParams class¶

Provides additional information about the acquired image.

int GetDataFormat()¶

Description: Returns data format of the last acquired image.

int GetWidth()¶

Description: Returns width of the last acquired image.

int GetHeight()¶

Description: Returns height of the last acquired image.

int GetFrameNum()¶

Description: Returns number of the last acquired image.

double GetTimestamp()¶

Description: Returns time stamp in seconds with microsecond precision.

Note: On most cameras the TimeStamp is represented in 32 bit microsecond number. The result resets automatically to zero after 2^32 micro-seconds (precisely 4294 seconds and 967296 micro seconds).

int GetDigitalInputLevel()¶

Description: Returns GPI input level.

int GetBlackLevel()¶

Description: Returns image black level (only for RAW and Mono formats).

xiCamEnum class¶

Contains function definitions for fast camera enumeration and identification.

int ReEnumerate()¶

Description: Re-enumerate available cameras.

Return: Returns the number of currently available devices.

string GetSerialNumById(int devIndex)¶

Description: Retrieve serial number of device, in string format.

Parameters:

int devIndex: Index of device to be queried.

Return: String format of camera serial number to be returned.

string GetDevNameById(int devIndex)¶

Description: Retrieve model name of device, in string format.

Parameters:

int devIndex: Index of device to be queried.

Return: String format of camera model name to be returned.

string GetInstancePathById(int devIndex)¶

Description: Retrieve system instance path of device, in string format.

Parameters:

int devIndex: Index of device to be queried.

Return: String format of system instance path of device to be returned.

string GetDeviceTypeById(int devIndex)¶

Description: Retrieve type of device, in string format.

Parameters:

int devIndex: Index of device to be queried.

Return: String format of device type to be returned.

string GetDeviceUserIdById(int devIndex)¶

Description: Retrieve custom device ID, in string format.

Parameters:

int devIndex: Index of device to be queried.

Return: String format of custom device ID type to be returned.

xiAPI.NET 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¶

PRM.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:

```
cam.SetParam(PRM.EXPOSURE, time_in_us);
```

PRM.EXPOSURE_TIME_SELECTOR¶

Type: Enumerator.

Default value: EXPOSURE_TIME_SELECTOR_COMMON

Usage:

```
xiCam.SetParam(PRM.EXPOSURE_TIME_SELECTOR, int val);
xiCam.GetParam(PRM.EXPOSURE_TIME_SELECTOR, out int val);
```

Value	Description
EXPOSURE_TIME_SELECTOR_COMMON	Selects the common Exposure Time
EXPOSURE_TIME_SELECTOR_GROUP1	Selects the common Exposure Time for pixel group 1 (for InterlineExposureMode)
EXPOSURE_TIME_SELECTOR_GROUP2	Selects the common Exposure Time for pixel group 2 (for InterlineExposureMode)

EXPOSURE_TIME_SELECTOR_DUAL_TRG_EXP_ZONE_1	Selects the Exposure Time for Zone 1 (for Dual Trigger Exposure feature)
EXPOSURE_TIME_SELECTOR_DUAL_TRG_EXP_ZONE_2	Selects the Exposure Time for Zone 2 (for Dual Trigger Exposure feature)

PRM.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:

```
xiCam.SetParam(PRM.EXPOSURE_BURST_COUNT, int val);
xiCam.GetParam(PRM.EXPOSURE_BURST_COUNT, out int val);
```

PRM.GAIN_SELECTOR¶

Description: Selects type of gain for GAIN . On some cameras there is possibility to select analog or digital gain separately.

Selector GAIN_SELECTOR_ALL is mapped on most cameras to analog gain.

Type: Enumerator.

Default value: GAIN_SELECTOR_ANALOG_ALL

Usage:

```
xiCam.SetParam(PRM.GAIN_SELECTOR, int val);
xiCam.GetParam(PRM.GAIN_SELECTOR, out int val);
```

Value	Description
GAIN_SELECTOR_ALL	Gain selector selects all channels. Implementation of gain type depends on camera.
GAIN_SELECTOR_ANALOG_ALL	Gain selector selects all analog channels. This is available only on some cameras.
GAIN_SELECTOR_DIGITAL_ALL	Gain selector selects all digital channels. This is available only on some cameras.

GAIN_SELECTOR_ANALOG_TAP1	Gain selector selects tap 1. This is available only on some cameras.
GAIN_SELECTOR_ANALOG_TAP2	Gain selector selects tap 2. This is available only on some cameras.
GAIN_SELECTOR_ANALOG_TAP3	Gain selector selects tap 3. This is available only on some cameras.
GAIN_SELECTOR_ANALOG_TAP4	Gain selector selects tap 4. This is available only on some cameras.
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.
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.

PRM.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: DOWNSAMPLING_TYPE, DOWNSAMPLING, BINNING_VERTICAL, BINNING_HORIZONTAL, DECIMATION_VERTICAL, DECIMATION_HORIZONTAL, DP_PARAM_VALUE, HDR, SENSOR_DATA_BIT_DEPTH, USER_SET_LOAD

Usage:

```
xiCam.SetParam(PRM.GAIN, float val);
xiCam.GetParam(PRM.GAIN, out float val);
```

PRM.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:

Usage:

```
xiCam.SetParam(PRM.DOWNSAMPLING, int val);
xiCam.GetParam(PRM.DOWNSAMPLING, out int val);
```

Value	Description
DWN_1x1	1 sensor pixel = 1 image pixel
DWN_2x2	2x2 sensor pixels = 1 image pixel
DWN_3x3	Downsampling 3x3.
DWN_4x4	4x4 sensor pixels = 1 image pixel
DWN_5x5	Downsampling 5x5.
DWN_6x6	Downsampling 6x6.
DWN_7x7	Downsampling 7x7.
DWN_8x8	Downsampling 8x8.
DWN_9x9	Downsampling 9x9.
DWN_10x10	Downsampling 10x10.
DWN_16x16	Downsampling 16x16.

PRM.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: BINNING

Usage:

```
xiCam.SetParam(PRM.DOWNSAMPLING_TYPE, int val);
xiCam.GetParam(PRM.DOWNSAMPLING_TYPE, out int val);
```

Value	Description
BINNING	pixels are interpolated - better image
SKIPPING	pixels are skipped - higher frame rate

PRM.TEST_PATTERN_GENERATOR_SELECTOR¶

Description: Selects Test Pattern Generator Engine.

Type: Enumerator.

Default value: SENSOR

Usage:

```
xiCam.SetParam(PRM.TEST_PATTERN_GENERATOR_SELECTOR, int val);
xiCam.GetParam(PRM.TEST_PATTERN_GENERATOR_SELECTOR, out int val);
```

Value	Description
SENSOR	Sensor test pattern generator
FPGA	FPGA Test Pattern Generator
MCU	MCU Test Pattern Generator

PRM.TEST_PATTERN¶

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

Type: Enumerator.

Default value: OFF

Is invalidated by: TEST_PATTERN_GENERATOR_SELECTOR

Usage:

```
xiCam.SetParam(PRM.TEST_PATTERN, int val);
xiCam.GetParam(PRM.TEST_PATTERN, out int val);
```

Value	Description
OFF	Testpattern turned off.
BLACK	Image is filled with darkest possible image.
WHITE	Image is filled with brightest possible image.
GREY_HORIZ_RAMP	Image is filled horizontally with an image that goes from the darkest possible value to the brightest.
GREY_VERT_RAMP	Image is filled vertically with an image that goes from the darkest possible value to the brightest.
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.
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.
HORIZ_LINE_MOVING	A moving horizontal line is superimposed on the live image.
VERT_LINE_MOVING	A moving vertical line is superimposed on the live image.

COLOR_BAR	Image is filled with stripes of color including White, Black, Red, Green, Blue, Cyan, Magenta and Yellow.
FRAME_COUNTER	A frame counter is superimposed on the live image.
DEVICE_SPEC_COUNTER	128bit counter.

PRM.IMAGE_DATA_FORMAT¶

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 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 IMAGE_DATA_FORMAT :

Format: MONO8 **Parameters controlled automatically:**

- SENSOR_DATA_BIT_DEPTH = 8 (see Note1)
- OUTPUT_DATA_BIT_DEPTH = 8 (see Note1)
- OUTPUT_DATA_PACKING = OFF

Image Processing: enabled

Format: RAW8 **Parameters controlled automatically:**

- SENSOR_DATA_BIT_DEPTH = 8 (see Note1)
- OUTPUT_DATA_BIT_DEPTH = 8 (see Note1)
- OUTPUT_DATA_PACKING = OFF

Image Processing: disabled

Format: MONO16 (see Note2) **Parameters controlled automatically:**

- SENSOR_DATA_BIT_DEPTH = maximum
- OUTPUT_DATA_BIT_DEPTH = SENSOR_DATA_BIT_DEPTH
- OUTPUT_DATA_PACKING = ON (see Note1)

Image Processing: enabled

Format: RAW16 (see Note2) **Parameters controlled automatically:**

- SENSOR_DATA_BIT_DEPTH = maximum
- OUTPUT_DATA_BIT_DEPTH = SENSOR_DATA_BIT_DEPTH

- OUTPUT_DATA_PACKING = ON (see Note1)

Image Processing: disabled
Format: RGB32, RGB24, RGBPLANAR **Parameters controlled automatically:**

- SENSOR_DATA_BIT_DEPTH = maximum
- OUTPUT_DATA_BIT_DEPTH = SENSOR_DATA_BIT_DEPTH
- OUTPUT_DATA_PACKING = ON (see Note1)

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

Note2: For RAW16, MONO16 the parameter IMAGE_DATA_BIT_DEPTH will be equal to OUTPUT_DATA_BIT_DEPTH . For other formats the IMAGE_DATA_BIT_DEPTH will be 8.

After changing of 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 RAW16bit is selected - each 16bit word (pixel) can contain values in range 0-1023.

Note4: For color modes RGB32 and 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. RGB24 is being processed from the RGB32 by removing the unused Alpha channel creating a slightly higher CPU load then the RGB32 format.

Type: Enumerator.

Default value: MONO8
Usage:

```
xiCam.SetParam(PRM.IMAGE_DATA_FORMAT, int val);
xiCam.GetParam(PRM.IMAGE_DATA_FORMAT, out int val);
```

Value	Description
MONO8	8 bits per pixel. [Intensity] (see Note5,Note6)
MONO16	16 bits per pixel. [Intensity LSB] [Intensity MSB] (see Note5,Note6)
RGB24	RGB data format. [Blue][Green][Red] (see Note5)
RGB32	RGBA data format. [Blue][Green][Red][0] (see Note5)

RGBPLANAR	RGB planar data format. [Red][Red]...[Green][Green]...[Blue][Blue]... (see Note5)
RAW8	8 bits per pixel raw data from sensor. [pixel byte] raw data from transport (camera output)
RAW16	16 bits per pixel raw data from sensor. [pixel byte low] [pixel byte high] 16 bits (depacked) raw data
FRM_TRANSPORT_DATA	Data from transport layer (e.g. packed). Depends on data on the transport layer (see Note7)
RGB48	RGB data format. [Blue low byte][Blue high byte][Green low][Green high][Red low][Red high] (see Note5)
RGB64	RGBA data format. [Blue low byte][Blue high byte][Green low][Green high][Red low][Red high][0][0] (Note5)
RGB16_PLANAR	RGB16 planar data format
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)
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)
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)
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_FLOAT32	32 bits per pixel raw data from sensor in single-precision floating point format. 4 bytes per pixel (depacked) raw data
RAW8X3	8 bits per pixel raw data from sensor(3 components in a row). [ch1 pixel byte] [ch2 pixel byte] [ch3 pixel byte] raw data from 3 channels (e.g. ToF cameras)
RAW16X3	16 bits per pixel raw data from sensor(3 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] 16 bits (depacked) raw data from 3 channels (e.g. ToF cameras)

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.

PRM.IMAGE_DATA_SIGN¶
Description: Signedness of image data.

Type: Enumerator.

Default value: DATA_SM_UNSIGNED
Is invalidated by: IMAGE_DATA_FORMAT, TOF_READOUT_MODE
Usage:

```
xiCam.GetParam(PRM.IMAGE_DATA_SIGN, out int val);
```

Value	Description
DATA_SM_UNSIGNED	Unsigned if it can only represent non-negative numbers (zero or positive numbers).
DATA_SM_SIGNED_2C	Signed if it can represent both positive and negative numbers (two's complement).
DATA_SM_SIGNED_FLOATING	Signed floating point data type.

PRM.SHUTTER_TYPE¶
Description: Type of sensor shutter.

Type: Enumerator.

Default value: SHUTTER_GLOBAL
Is invalidated by: TRG_SOURCE
Usage:

```
xiCam.SetParam(PRM.SHUTTER_TYPE, int val);
xiCam.GetParam(PRM.SHUTTER_TYPE, out int val);
```

Value	Description
SHUTTER_GLOBAL	Sensor Global Shutter(CMOS sensor)
SHUTTER_ROLLING	Sensor Electronic Rolling Shutter(CMOS sensor)

SHUTTER_GLOBAL_RESET_RELEASE	Sensor Global Reset Release Shutter(CMOS sensor)

PRM.SENSOR_TAPS¶

Description: Set/get the number of taps used on sensor.

Type: Enumerator.

Default value:

Usage:

```
xiCam.SetParam(PRM.SENSOR_TAPS, int val);
xiCam.GetParam(PRM.SENSOR_TAPS, out int val);
```

Value	Description
TAP_CNT_1	1 sensor tap selected.
TAP_CNT_2	2 sensor taps selected.
TAP_CNT_4	4 sensor taps selected.

PRM.AEAG¶

Description: Automatic exposure/gain.

Type: Integer.

Default value: 0

Usage:

```
xiCam.SetParam(PRM.AEAG, int val);
xiCam.GetParam(PRM.AEAG, out int val);
```

PRM.AEAG_ROI_OFFSET_X¶

Description: X offset of the area used for AEAG calculation. The sum of AEAG_ROI_OFFSET_X and AEAG_ROI_WIDTH must be equal or lower than the image resolution(width).

Type: Integer.

Default value: 0

Usage:

```
xiCam.SetParam(PRM.AEAG_ROI_OFFSET_X, int val);
xiCam.GetParam(PRM.AEAG_ROI_OFFSET_X, out int val);
```

PRM.AEAG_ROI_OFFSET_Y¶

Description: Y offset of the area used for AEAG calculation. The sum of AEAG_ROI_OFFSET_Y and AEAG_ROI_HEIGHT must be equal or lower than the image resolution(height).

Type: Integer.

Default value: 0
Usage:

```
xiCam.SetParam(PRM.AEAG_ROI_OFFSET_Y, int val);
xiCam.GetParam(PRM.AEAG_ROI_OFFSET_Y, out int val);
```

PRM.AEAG_ROI_WIDTH¶
Description: width of the area used for AEAG calculation. The sum of AEAG_ROI_OFFSET_X and 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:

```
xiCam.SetParam(PRM.AEAG_ROI_WIDTH, int val);
xiCam.GetParam(PRM.AEAG_ROI_WIDTH, out int val);
```

PRM.AEAG_ROI_HEIGHT¶
Description: height of the area used for AEAG calculation. The sum of AEAG_ROI_OFFSET_Y and 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:

```
xiCam.SetParam(PRM.AEAG_ROI_HEIGHT, int val);
xiCam.GetParam(PRM.AEAG_ROI_HEIGHT, out int val);
```

PRM.SENS_DEFECTS_CORR_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:

```
xiCam.SetParam(PRM.SENS_DEFECTS_CORR_LIST_SELECTOR, int val);
xiCam.GetParam(PRM.SENS_DEFECTS_CORR_LIST_SELECTOR, out int val);
```

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

PRM.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:

```
xiCam.SetParam(PRM.SENS_DEFECTS_CORR_LIST_CONTENT, string val);
xiCam.GetParam(PRM.SENS_DEFECTS_CORR_LIST_CONTENT, out string val);
```

PRM.SENS_DEFECTS_CORR¶

Description: Correction of sensor defects. For more information see Sensor Defect Correction support page.

Type: Integer.

Default value: 0

Usage:

```
xiCam.SetParam(PRM.SENS_DEFECTS_CORR, int val);
xiCam.GetParam(PRM.SENS_DEFECTS_CORR, out int val);
```

PRM.AUTO_WB¶

Description: Automatic white balance.

Type: Integer.

Default value: 0

Usage:

```
xiCam.SetParam(PRM.AUTO_WB, int val);
xiCam.GetParam(PRM.AUTO_WB, out int val);
```

PRM.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:

Default value: 0
Usage:

```
xiCam.SetParam(PRM.MANUAL_WB,  val);
```

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

Supported cameras: MC,CB,MX

Type: Integer.

Default value: 0
Usage:

```
xiCam.SetParam(PRM.WB_ROI_OFFSET_X, int val);
xiCam.GetParam(PRM.WB_ROI_OFFSET_X, out int val);
```

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

Supported cameras: MC,CB,MX

Type: Integer.

Default value: 0
Usage:

```
xiCam.SetParam(PRM.WB_ROI_OFFSET_Y, int val);
xiCam.GetParam(PRM.WB_ROI_OFFSET_Y, out int val);
```

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

Supported cameras: MC,CB,MX

Type: Integer.

Default value: 0
Usage:

```
xiCam.SetParam(PRM.WB_ROI_WIDTH, int val);
xiCam.GetParam(PRM.WB_ROI_WIDTH, out int val);
```

PRM.WB_ROI_HEIGHT¶

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

Supported cameras: MC,CB,MX

Type: Integer.

Default value: 0

Usage:

```
xiCam.SetParam(PRM.WB_ROI_HEIGHT, int val);
xiCam.GetParam(PRM.WB_ROI_HEIGHT, out int val);
```

PRM.WB_KR¶

Description: White balance red coefficient.

Type: Float.

Default value: 1.0

Typical range: [0.0, 10.0]

Usage:

```
xiCam.SetParam(PRM.WB_KR, float val);
xiCam.GetParam(PRM.WB_KR, out float val);
```

PRM.WB_KG¶

Description: White balance green coefficient.

Type: Float.

Default value: 1.0

Typical range: [0.0, 10.0]

Usage:

```
xiCam.SetParam(PRM.WB_KG, float val);
xiCam.GetParam(PRM.WB_KG, out float val);
```

PRM.WB_KB¶

Description: White balance blue coefficient.

Type: Float.

Default value: 1.0

Typical range: [0.0, 10.0]

Usage:

```
xiCam.SetParam(PRM.WB_KB, float val);
```

```
xiCam.GetParam(PRM.WB_KB, out float val);
```

PRM.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 OFFSET_X and 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 INCREMENT . If camera runs in multiple region mode (REGION_SELECTOR) this parameter is width of region currently selected (in pixels).

Type: Integer.

Default value: Full resolution width.
Is invalidated by: BINNING_HORIZONTAL, DECIMATION_HORIZONTAL, DOWNSAMPLING_TYPE, DOWNSAMPLING, IMAGE_DATA_FORMAT, IMAGE_AREA, OUTPUT_DATA_PACKING, OUTPUT_DATA_BIT_DEPTH, HEIGHT
Usage:

```
xiCam.SetParam(PRM.WIDTH, int val);
xiCam.GetParam(PRM.WIDTH, out int val);
```

PRM.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 OFFSET_Y and 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 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: BINNING_VERTICAL, DECIMATION_VERTICAL, DOWNSAMPLING_TYPE, DOWNSAMPLING, IMAGE_DATA_FORMAT, OUTPUT_DATA_PACKING, IMAGE_AREA, OUTPUT_DATA_BIT_DEPTH, WIDTH
Usage:

```
xiCam.SetParam(PRM.HEIGHT, int val);
xiCam.GetParam(PRM.HEIGHT, out int val);
```

PRM.OFFSET_X¶

Description: Horizontal offset from the origin to the area of interest (in pixels). The sum of OFFSET_X and 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 INCREMENT .

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

Type: Integer.

Default value: 0
Is invalidated by: BINNING_HORIZONTAL, DECIMATION_HORIZONTAL, DOWNSAMPLING_TYPE, DOWNSAMPLING, IMAGE_AREA
Usage:

```
xiCam.SetParam(PRM.OFFSET_X, int val);
xiCam.GetParam(PRM.OFFSET_X, out int val);
```

PRM.OFFSET_Y¶
Description: Vertical offset from the origin to the area of interest (in pixels).The sum of OFFSET_Y and 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 INCREMENT .

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

Type: Integer.

Default value: 0
Is invalidated by: BINNING_VERTICAL, DECIMATION_VERTICAL, DOWNSAMPLING_TYPE, DOWNSAMPLING, IMAGE_AREA
Usage:

```
xiCam.SetParam(PRM.OFFSET_Y, int val);
xiCam.GetParam(PRM.OFFSET_Y, out int val);
```

PRM.REGION_SELECTOR¶
Description: Selects Region in Multiple ROI. Parameters: WIDTH , HEIGHT , OFFSET_X , OFFSET_Y , 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
Typical range: [0, 15]
Is invalidated by: BINNING_VERTICAL, BINNING_HORIZONTAL, DECIMATION_VERTICAL, DECIMATION_HORIZONTAL, DOWNSAMPLING_TYPE, DOWNSAMPLING
Usage:

```
xiCam.SetParam(PRM.REGION_SELECTOR, int val);
xiCam.GetParam(PRM.REGION_SELECTOR, out int val);
```

PRM.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: BINNING_VERTICAL, BINNING_HORIZONTAL, DECIMATION_VERTICAL, DECIMATION_HORIZONTAL, DOWNSAMPLING_TYPE, DOWNSAMPLING
Usage:

```
xiCam.SetParam(PRM.REGION_MODE, int val);
xiCam.GetParam(PRM.REGION_MODE, out int val);
```

PRM.REGIONS_GEOMETRY¶

Description: Type of multiple regions geometry.

Type: Enumerator.

Default value:
Usage:

```
xiCam.GetParam(PRM.REGIONS_GEOMETRY, out int val);
```

Value	Description
REGIONS_GEOMETRY_SINGLE_REGION_ONLY	Single region
REGIONS_GEOMETRY_MULTIPLE_REGIONS_IN_SINGLE_COLUMN	Multiple regions in single column
REGIONS_GEOMETRY_MULTIPLE_REGIONS_IN_MULTIPLE_COLUMNS	Multiple regions in multiple columns

PRM.HORIZONTAL_FLIP¶

Description: Activates horizontal flip if available in camera.

Type: Integer.

Default value: 0 for disabled flipping.
Usage:

```
xiCam.SetParam(PRM.HORIZONTAL_FLIP, int val);
xiCam.GetParam(PRM.HORIZONTAL_FLIP, out int val);
```

PRM.VERTICAL_FLIP¶

Description: Activates vertical flip if available in camera.

Type: Integer.

Default value: 0

Usage:

```
xiCam.SetParam(PRM.VERTICAL_FLIP, int val);
xiCam.GetParam(PRM.VERTICAL_FLIP, out int val);
```

PRM.INTERLINE_EXPOSURE_MODE¶

Description: Selector for Exposure parameter

Type: Enumerator.

Default value: INTERLINE_EXPOSURE_MODE_OFF

Usage:

```
xiCam.SetParam(PRM.INTERLINE_EXPOSURE_MODE, int val);
xiCam.GetParam(PRM.INTERLINE_EXPOSURE_MODE, out int val);
```

Value	Description
INTERLINE_EXPOSURE_MODE_OFF	Disabled
INTERLINE_EXPOSURE_MODE_ON	Enabled

PRM.FFC¶

Description: Image flat field correction.(NEW_PROCESS_CHAIN_ENABLE must be XI_ON). For more information see Flat Field Correction support page.

Type: Integer.

Default value: 0

Usage:

```
xiCam.SetParam(PRM.FFC, int val);
xiCam.GetParam(PRM.FFC, out int val);
```

PRM.FFC_FLAT_FIELD_FILE_NAME¶

Description: Set name of file of image flat field to be applied for FFC processor(in tiff format). (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:

```
xiCam.SetParam(PRM.FFC_FLAT_FIELD_FILE_NAME, string val);
xiCam.GetParam(PRM.FFC_FLAT_FIELD_FILE_NAME, out string val);
```

PRM.FFC_DARK_FIELD_FILE_NAME¶
Description: Set name of file of image dark field to be applied for FFC processor(in tiff format). (NEW_PROCESS_CHAIN_ENABLE must be XI_ON). For more information see Flat Field Correction support page.

Type: String.

Default value: -
Usage:

```
xiCam.SetParam(PRM.FFC_DARK_FIELD_FILE_NAME, string val);
xiCam.GetParam(PRM.FFC_DARK_FIELD_FILE_NAME, out string val);
```

PRM.TOF_READOUT_MODE¶
Description: Sets ToF Readout Mode

Type: Enumerator.

Default value: TOF_READOUT_MODE_A_ONLY
Usage:

```
xiCam.SetParam(PRM.TOF_READOUT_MODE, int val);
xiCam.GetParam(PRM.TOF_READOUT_MODE, out int val);
```

Value	Description
TOF_READOUT_MODE_A_ONLY	A Only readout mode
TOF_READOUT_MODE_B_ONLY	B Only readout mode
TOF_READOUT_MODE_A_MINUS_B	A Minus B readout mode
TOF_READOUT_MODE_A_PLUS_B	A Plus B readout mode
TOF_READOUT_MODE_A_AND_B	A And B readout mode

PRM.TOF_MODULATION_FREQUENCY¶
Description: Sets ToF Modulation Frequency in MHz

Type: Float.

Default value: 100.0

Typical range: [4.0, 100.0]
Usage:

```
xiCam.SetParam(PRM.TOF_MODULATION_FREQUENCY, float val);
xiCam.GetParam(PRM.TOF_MODULATION_FREQUENCY, out float val);
```

PRM.TOF_MULTIPLE_PHASES_IN_BUFFER¶
Description: is multiple ToF phases concatenated in buffer

Type: Integer.

Default value: 0
Usage:

```
xiCam.GetParam(PRM.TOF_MULTIPLE_PHASES_IN_BUFFER, out int val);
```

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

Type: Integer.

Default value: 1
Usage:

```
xiCam.SetParam(PRM.TOF_PHASES_COUNT, int val);
xiCam.GetParam(PRM.TOF_PHASES_COUNT, out int val);
```

PRM.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: TOF_PHASE_SELECTOR
Usage:

```
xiCam.SetParam(PRM.TOF_PHASE_ANGLE, float val);
xiCam.GetParam(PRM.TOF_PHASE_ANGLE, out float val);
```

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

Type: Float.

Default value: 1000.0
Is invalidated by: TOF_PHASE_SELECTOR, TOF_MODULATION_FREQUENCY
Usage:

```
xiCam.SetParam(PRM.TOF_PHASE_EXPOSURE_TIME, float val);
xiCam.GetParam(PRM.TOF_PHASE_EXPOSURE_TIME, out float val);
```

PRM.TOF_PHASE_SELECTOR¶
Description: Selects tof phase

Type: Integer.

Default value: 1
Usage:

```
xiCam.SetParam(PRM.TOF_PHASE_SELECTOR, int val);
xiCam.GetParam(PRM.TOF_PHASE_SELECTOR, out int val);
```

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. 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. 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. 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. BINNING_VERTICAL_MODE)

PRM.BINNING_SELECTOR¶
Description: Selects which binning engine is controlled by the BinningHorizontal and BinningVertical features.

Type: Enumerator.

Default value: SENSOR
Usage:

```
xiCam.SetParam(PRM.BINNING_SELECTOR, int val);
xiCam.GetParam(PRM.BINNING_SELECTOR, out int val);
```

Value	Description
SENSOR	parameters for image sensor binning are selected

DEVICE_FPGA	parameters for device (camera) FPGA decimation are selected
HOST_CPU	parameters for Host CPU binning are selected

PRM.BINNING_VERTICAL_MODE¶

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

Type: Enumerator.

Default value: SUM

Usage:

```
xiCam.SetParam(PRM.BINNING_VERTICAL_MODE, int val);
xiCam.GetParam(PRM.BINNING_VERTICAL_MODE, out int val);
```

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

PRM.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: BINNING_SELECTOR, DOWNSAMPLING_TYPE, DOWNSAMPLING, SHUTTER_TYPE, DECIMATION_VERTICAL, DECIMATION_HORIZONTAL, BINNING_HORIZONTAL, DP_UNIT_SELECTOR, DP_PROC_SELECTOR, DP_PARAM_SELECTOR, DP_PARAM_VALUE, HDR

Usage:

```
xiCam.SetParam(PRM.BINNING_VERTICAL, int val);
xiCam.GetParam(PRM.BINNING_VERTICAL, out int val);
```

PRM.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: BINNING_SELECTOR, DOWNSAMPLING_TYPE, DOWNSAMPLING, DECIMATION_VERTICAL, DECIMATION_HORIZONTAL, BINNING_HORIZONTAL, DP_UNIT_SELECTOR, DP_PROC_SELECTOR, DP_PARAM_SELECTOR, DP_PARAM_VALUE, HDR
Usage:

```
xiCam.SetParam(PRM.BINNING_VERTICAL_FLOAT, float val);
xiCam.GetParam(PRM.BINNING_VERTICAL_FLOAT, out float val);
```

PRM.BINNING_HORIZONTAL_MODE¶
Description: Sets the mode to use to combine horizontal photo-sensitive cells together when BinningHorizontal is used.

Type: Enumerator.

Default value: SUM
Usage:

```
xiCam.SetParam(PRM.BINNING_HORIZONTAL_MODE, int val);
xiCam.GetParam(PRM.BINNING_HORIZONTAL_MODE, out int val);
```

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

PRM.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: BINNING_SELECTOR, DOWNSAMPLING_TYPE, DOWNSAMPLING, SHUTTER_TYPE, DECIMATION_VERTICAL, DECIMATION_HORIZONTAL, BINNING_VERTICAL, DP_UNIT_SELECTOR, DP_PROC_SELECTOR, DP_PARAM_SELECTOR, DP_PARAM_VALUE

Usage:

```
xiCam.SetParam(PRM.BINNING_HORIZONTAL, int val);
xiCam.GetParam(PRM.BINNING_HORIZONTAL, out int val);
```

PRM.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: BINNING_SELECTOR, DOWNSAMPLING_TYPE, DOWNSAMPLING, DECIMATION_VERTICAL, DECIMATION_HORIZONTAL, BINNING_VERTICAL, DP_UNIT_SELECTOR, DP_PROC_SELECTOR, DP_PARAM_SELECTOR, DP_PARAM_VALUE
Usage:

```
xiCam.SetParam(PRM.BINNING_HORIZONTAL_FLOAT, float val);
xiCam.GetParam(PRM.BINNING_HORIZONTAL_FLOAT, out float val);
```

PRM.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: MONO
Usage:

```
xiCam.SetParam(PRM.BINNING_HORIZONTAL_PATTERN, int val);
xiCam.GetParam(PRM.BINNING_HORIZONTAL_PATTERN, out int val);
```

Value	Description
MONO	adjacent pixels are combined
BAYER	Bayer pattern is preserved during pixel combining

PRM.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: MONO
Usage:

```
xiCam.SetParam(PRM.BINNING_VERTICAL_PATTERN, int val);
xiCam.GetParam(PRM.BINNING_VERTICAL_PATTERN, out int val);
```

Value	Description
MONO	adjacent pixels are combined
BAYER	Bayer pattern is preserved during pixel combining

PRM.DECIMATION_SELECTOR¶
Description: Selects Decimation engine to configure.

Type: Enumerator.

Default value: SENSOR
Usage:

```
xiCam.SetParam(PRM.DECIMATION_SELECTOR, int val);
xiCam.GetParam(PRM.DECIMATION_SELECTOR, out int val);
```

Value	Description
SENSOR	parameters for image sensor decimation are selected
DEVICE_FPGA	parameters for device (camera) FPGA decimation are selected
HOST_CPU	parameters for Host CPU decimation are selected

PRM.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: DECIMATION_VERTICAL, DOWNSAMPLING_TYPE, DOWNSAMPLING, BINNING_VERTICAL, BINNING_HORIZONTAL, DECIMATION_HORIZONTAL, DP_UNIT_SELECTOR, DP_PROC_SELECTOR, DP_PARAM_SELECTOR, DP_PARAM_VALUE, DECIMATION_SELECTOR, HDR

Usage:

```
xiCam.SetParam(PRM.DECIMATION_VERTICAL, int val);
xiCam.GetParam(PRM.DECIMATION_VERTICAL, out int val);
```

PRM.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: DECIMATION_VERTICAL, DOWNSAMPLING_TYPE, DOWNSAMPLING, BINNING_VERTICAL, BINNING_HORIZONTAL, DECIMATION_VERTICAL, DP_UNIT_SELECTOR, DP_PROC_SELECTOR, DP_PARAM_SELECTOR, DP_PARAM_VALUE, DECIMATION_SELECTOR

Usage:

```
xiCam.SetParam(PRM.DECIMATION_HORIZONTAL, int val);
xiCam.GetParam(PRM.DECIMATION_HORIZONTAL, out int val);
```

PRM.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: MONO

Usage:

```
xiCam.SetParam(PRM.DECIMATION_HORIZONTAL_PATTERN, int val);
xiCam.GetParam(PRM.DECIMATION_HORIZONTAL_PATTERN, out int val);
```

Value	Description
MONO	adjacent pixels are decimated
BAYER	Bayer pattern is preserved during pixel decimation

PRM.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: MONO
Usage:

```
xiCam.SetParam(PRM.DECIMATION_VERTICAL_PATTERN, int val);
xiCam.GetParam(PRM.DECIMATION_VERTICAL_PATTERN, out int val);
```

Value	Description
MONO	adjacent pixels are decimated
BAYER	Bayer pattern is preserved during pixel decimation

AE Setup¶

PRM.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:

```
xiCam.SetParam(PRM.EXP_PRIORITY, float val);
xiCam.GetParam(PRM.EXP_PRIORITY, out float val);
```

PRM.AG_MAX_LIMIT¶

Description: Maximum limit of gain in AEAG procedure.

Type: Float.

Default value: Depends on camera type (dB).
Usage:

```
xiCam.SetParam(PRM.AG_MAX_LIMIT, float val);
xiCam.GetParam(PRM.AG_MAX_LIMIT, out float val);
```

PRM.AE_MAX_LIMIT¶

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

Type: Integer.

Default value: 200000
Typical range: [0, 1000000]
Usage:

```
xiCam.SetParam(PRM.AE_MAX_LIMIT, int val);
xiCam.GetParam(PRM.AE_MAX_LIMIT, out int val);
```

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

Type: Integer.

Default value: 50
Typical range: [0, 100]
Usage:

```
xiCam.SetParam(PRM.AEAG_LEVEL, int val);
xiCam.GetParam(PRM.AEAG_LEVEL, out int val);
```

PRM.AEAG_SKIP_FRAMES_COUNT¶
Description: Minimum number of frames to skip between AEAG compensation and AEAG re-evaluation. Typically, changes in Exposure or Gain compensation take effect after several frames. This parameter can tune the AEAG processor. A low count may cause oscillations, while a high count may result in a slow AEAG response.

Type: Integer.

Default value: 1
Typical range: [0, 100]
Usage:

```
xiCam.SetParam(PRM.AEAG_SKIP_FRAMES_COUNT, int val);
xiCam.GetParam(PRM.AEAG_SKIP_FRAMES_COUNT, out int val);
```

Performance¶

PRM.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 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:

```
xiCam.SetParam(PRM.LIMIT_BANDWIDTH, int val);
xiCam.GetParam(PRM.LIMIT_BANDWIDTH, out int val);
```

PRM.LIMIT_BANDWIDTH_MODE¶
Description: Controls if the LIMIT_BANDWIDTH is active. When disabled, lower level specific features are expected to control the throughput. When enabled, LIMIT_BANDWIDTH controls the overall throughput.

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

Type: Enumerator.

Default value: ON
Usage:

```
xiCam.SetParam(PRM.LIMIT_BANDWIDTH_MODE, int val);
xiCam.GetParam(PRM.LIMIT_BANDWIDTH_MODE, out int val);
```

Value	Description
OFF	Turn parameter off
ON	Turn parameter on

PRM.SENSOR_DATA_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:
Is invalidated by: IMAGE_DATA_FORMAT, USER_SET_LOAD, DUAL_ADC_MODE, DOWNSAMPLING
Usage:

```
xiCam.SetParam(PRM.SENSOR_DATA_BIT_DEPTH, int val);
xiCam.GetParam(PRM.SENSOR_DATA_BIT_DEPTH, out int val);
```

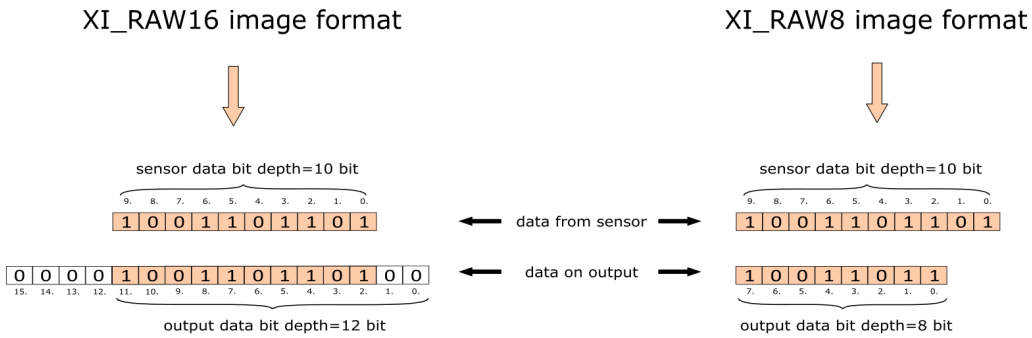
Value	Description
BPP_8	8 bit per pixel
BPP_9	9 bit per pixel
BPP_10	10 bit per pixel

BPP_11	11 bit per pixel
BPP_12	12 bit per pixel
BPP_13	13 bit per pixel
BPP_14	14 bit per pixel
BPP_15	15 bit per pixel
BPP_16	16 bit per pixel
BPP_24	24 bit per pixel
BPP_32	32 bit per pixel

PRM.OUTPUT_DATA_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:

Is invalidated by: IMAGE_DATA_FORMAT, DP_PARAM_VALUE, BINNING_VERTICAL, BINNING_HORIZONTAL, DECIMATION_VERTICAL, DECIMATION_HORIZONTAL, SHUTTER_TYPE

Usage:

```
xiCam.SetParam(PRM.OUTPUT_DATA_BIT_DEPTH, int val);
xiCam.GetParam(PRM.OUTPUT_DATA_BIT_DEPTH, out int val);
```

Value	Description
BPP_8	8 bit per pixel
BPP_9	9 bit per pixel
BPP_10	10 bit per pixel

BPP_11	11 bit per pixel
BPP_12	12 bit per pixel
BPP_13	13 bit per pixel
BPP_14	14 bit per pixel
BPP_15	15 bit per pixel
BPP_16	16 bit per pixel
BPP_24	24 bit per pixel
BPP_32	32 bit per pixel

PRM.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:

Is invalidated by: IMAGE_DATA_FORMAT

Usage:

```
xiCam.SetParam(PRM.IMAGE_DATA_BIT_DEPTH, int val);
xiCam.GetParam(PRM.IMAGE_DATA_BIT_DEPTH, out int val);
```

Value	Description
BPP_8	8 bit per pixel
BPP_9	9 bit per pixel
BPP_10	10 bit per pixel
BPP_11	11 bit per pixel
BPP_12	12 bit per pixel
BPP_13	13 bit per pixel
BPP_14	14 bit per pixel
BPP_15	15 bit per pixel
BPP_16	16 bit per pixel

BPP_24	24 bit per pixel
BPP_32	32 bit per pixel

PRM.OUTPUT_DATA_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: 0

Is invalidated by: IMAGE_DATA_FORMAT, OUTPUT_DATA_BIT_DEPTH, DP_PARAM_VALUE, BINNING_VERTICAL, BINNING_HORIZONTAL, DECIMATION_VERTICAL, DECIMATION_HORIZONTAL, SHUTTER_TYPE

Usage:

```
xiCam.SetParam(PRM.OUTPUT_DATA_PACKING, int val);
xiCam.GetParam(PRM.OUTPUT_DATA_PACKING, out int val);
```

PRM.OUTPUT_DATA_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_GROUPING

Usage:

```
xiCam.SetParam(PRM.OUTPUT_DATA_PACKING_TYPE, int val);
xiCam.GetParam(PRM.OUTPUT_DATA_PACKING_TYPE, out int val);
```

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

Temperature¶

PRM.IS_COOLED¶

Description: Returns 1 for cameras that support cooling.

Type: Integer.

Default value: 0

Usage:

```
xiCam.GetParam(PRM.IS_COOLED, out int val);
```

PRM.COOLING¶

Description: Set camera cooling control. Replaced by TEMP_CONTROL_MODE

Type: Enumerator.

Default value: XI_TEMP_CTRL_MODE_OFF

Usage:

```
xiCam.SetParam(PRM.COOLING, int val);
xiCam.GetParam(PRM.COOLING, out int val);
```

Value	Description
CTRL_MODE_OFF	Controlling of elements (TEC/Peltier, Fans) is turned off
CTRL_MODE_AUTO	Controlling of elements is performed automatically by API or camera in order to reach parameter TARGET_TEMP.
CTRL_MODE_MANUAL	Controlling of elements is done manually by application.

PRM.TARGET_TEMP¶

Description: Set target temperature for automatic temperature control.

Type: Float.

Default value: 0.0

Typical range: [0.0, 200.0]

Usage:

```
xiCam.SetParam(PRM.TARGET_TEMP, float val);
xiCam.GetParam(PRM.TARGET_TEMP, out float val);
```

PRM.TEMP_SELECTOR¶

Description: Temperature sensor selector.

Type: Enumerator.

Default value: TEMP_SENSOR_BOARD

Usage:

```
xiCam.SetParam(PRM.TEMP_SELECTOR, int val);
xiCam.GetParam(PRM.TEMP_SELECTOR, out int val);
```

Value	Description

TEMP_IMAGE_SENSOR_RAW	Image sensor die (non-calibrated)
TEMP_IMAGE_SENSOR	Image sensor die (calibrated)
TEMP_SENSOR_BOARD	Image sensor PCB
TEMP_INTERFACE_BOARD	Data interface PCB
TEMP_FRONT_HOUSING	Front part of camera housing
TEMP_BACK_HOUSING	Rear part of camera housing
TEMP_TEC1_COLD	TEC1 cold side temperature
TEMP_TEC1_HOT	TEC1 hot side temperature
TEMP_VCSEL_BOARD_A	VCSEL board temperature

PRM.TEMP¶

Description: Selected thermometer reading in degree Celsius. Thermometer can be selected by TEMP_SELECTOR .

Type: Float.

Default value: 0.0
Is invalidated by: TEMP_SELECTOR
Usage:

```
xiCam.GetParam(PRM.TEMP, out float val);
```

PRM.TEMP_CONTROL_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: CTRL_MODE_OFF
Usage:

```
xiCam.SetParam(PRM.TEMP_CONTROL_MODE, int val);
xiCam.GetParam(PRM.TEMP_CONTROL_MODE, out int val);
```

Value	Description

CTRL_MODE_OFF	Controlling of elements (TEC/Peltier, Fans) is turned off
CTRL_MODE_AUTO	Controlling of elements is performed automatically by API or camera in order to reach parameter TARGET_TEMP.
CTRL_MODE_MANUAL	Controlling of elements is done manually by application.

PRM.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:

```
xiCam.GetParam(PRM.CHIP_TEMP, out float val);
```

PRM.HOUS_TEMP¶

Description: Camera housing temperature.

Type: Float.

Default value: 0.0

Usage:

```
xiCam.GetParam(PRM.HOUS_TEMP, out float val);
```

PRM.HOUS_BACK_SIDE_TEMP¶

Description: Camera housing back side temperature.

Type: Float.

Default value: 0.0

Usage:

```
xiCam.GetParam(PRM.HOUS_BACK_SIDE_TEMP, out float val);
```

PRM.SENSOR_BOARD_TEMP¶

Description: Camera sensor board temperature.

Type: Float.

Default value: 0.0

Usage:

```
xiCam.GetParam(PRM.SENSOR_BOARD_TEMP, out float val);
```

PRM.TEMP_ELEMENT_SEL¶

Description: Temperature element selector (TEC, Fan)

Type: Enumerator.

Default value: XI_TEMP_ELEM_TEC1

Usage:

```
xiCam.SetParam(PRM.TEMP_ELEMENT_SEL, int val);
xiCam.GetParam(PRM.TEMP_ELEMENT_SEL, out int val);
```

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

PRM.TEMP_ELEMENT_VALUE¶

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: TEMP_ELEMENT_SEL

Usage:

```
xiCam.SetParam(PRM.TEMP_ELEMENT_VALUE, float val);
xiCam.GetParam(PRM.TEMP_ELEMENT_VALUE, out float val);
```

Color Correction¶

Note: Works only for color cameras.

PRM.CMS¶

Description: Enable or disable color management.

Note: This feature is in Beta stage.

Type: Enumerator.

Default value: DIS

Usage:

```
xiCam.SetParam(PRM.CMS, int val);
xiCam.GetParam(PRM.CMS, out int val);
```

Value	Description
DIS	disables color management
EN	enables color management (high CPU usage)
EN_FAST	enables fast color management (high RAM usage)

PRM.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: INTENT_PERCEPTUAL

Usage:

```
xiCam.SetParam(PRM.CMS_INTENT, int val);
xiCam.GetParam(PRM.CMS_INTENT, out int val);
```

Value	Description
INTENT_PERCEPTUAL	CMS intent perceptual
RELATIVE_COLORIMETRIC	CMS intent relative colorimetry
SATURATION	CMS intent saturation
ABSOLUTE_COLORIMETRIC	CMS intent absolute colorimetry

PRM.APPLY_CMS¶

Description: If set to XI_ON applies CMS profile to xiGetImage.

Type: Integer.

Default value: 0

Usage:

```
xiCam.SetParam(PRM.APPLY_CMS, int val);
xiCam.GetParam(PRM.APPLY_CMS, out int val);
```

PRM.INPUT_CMS_PROFILE¶

Description: Filename of the input cms profile (e.g. input.icc)

Type: String.

Default value: (default device profile)

Usage:

```
xiCam.SetParam(PRM.INPUT_CMS_PROFILE, string val);
```

```
xiCam.GetParam(PRM.INPUT_CMS_PROFILE, out string val);
```

PRM.OUTPUT_CMS_PROFILE¶

Description: Filename of the output cms profile (e.g. output.icc)

Type: String.

Default value: (sRGB profile)

Usage:

```
xiCam.SetParam(PRM.OUTPUT_CMS_PROFILE, string val);
xiCam.GetParam(PRM.OUTPUT_CMS_PROFILE, out string val);
```

PRM.IMAGE_IS_COLOR¶

Description: Returns 1 for color cameras.

Type: Integer.

Default value: 0

Usage:

```
xiCam.GetParam(PRM.IMAGE_IS_COLOR, out int val);
```

PRM.COLOR_FILTER_ARRAY¶

Description: Returns color filter array type of RAW data.

Type: Enumerator.

Default value: NONE

Usage:

```
xiCam.GetParam(PRM.COLOR_FILTER_ARRAY, out int val);
```

Value	Description
NONE	Result pixels have no filters applied in this format
BAYER_RGGB	Regular RGGB
CMYG	AK Sony sens
RGR	2R+G readout
BAYER_BGGR	BGGR readout
BAYER_GRBG	GRBG readout
BAYER_GBRG	GBRG readout
POLAR_A_BAYER_BGGR	BGGR polarized 4x4 macropixel

POLAR_A	Polarized 2x2 macropixel
TOF_ANB	ToF A and B
TOF_AMB	ToF A minus B
TOF_APB	ToF A plus B
TOF_A	ToF A only
TOF_B	ToF B only
TOF_X1X2X3	ToF X1,X2,X3

PRM.GAMMAY¶

Description: Luminosity gamma.

Lowering the value increases correction.

Type: Float.

Default value: 0.47
Typical range: [0.3, 1.0]
Usage:

```
xiCam.SetParam(PRM.GAMMAY, float val);
xiCam.GetParam(PRM.GAMMAY, out float val);
```

PRM.GAMMAC¶

Description: Chromaticity gamma.

Type: Float.

Default value: 0.8
Typical range: [0.0, 1.0]
Usage:

```
xiCam.SetParam(PRM.GAMMAC, float val);
xiCam.GetParam(PRM.GAMMAC, out float val);
```

PRM.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:

```
xiCam.SetParam(PRM.SHARPNESS, float val);
xiCam.GetParam(PRM.SHARPNESS, out float val);
```

PRM.CC_MATRIX_00¶

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:

```
xiCam.SetParam(PRM.CC_MATRIX_00, float val);
xiCam.GetParam(PRM.CC_MATRIX_00, out float val);
```

PRM.DEFAULT_CC_MATRIX¶

Description: Set default Color Correction Matrix

Type:

Default value: 0
Usage:

```
xiCam.SetParam(PRM.DEFAULT_CC_MATRIX,  val);
```

PRM.CC_MATRIX_NORM¶

Description: Activates normalization of color correction matrix.

Type: Integer.

Default value: 0 for disabled normalization.
Usage:

```
xiCam.SetParam(PRM.CC_MATRIX_NORM, int val);
xiCam.GetParam(PRM.CC_MATRIX_NORM, out int val);
```

Device IO¶

PRM.TRG_SOURCE¶

Description: Defines source of trigger.

Note: To set input as external trigger, GPI_MODE of selected input should be set to TRIGGER. See example at GPI_MODE .

Type: Enumerator.

Default value: OFF

Usage:

```
xiCam.SetParam(PRM.TRG_SOURCE, int val);
xiCam.GetParam(PRM.TRG_SOURCE, out int val);
```

Value	Description
OFF	Capture of next image is automatically started after previous.
EDGE_RISING	Capture is started on rising edge of selected input.
EDGE_FALLING	Capture is started on falling edge of selected input
SOFTWARE	Capture is started with software trigger.
LEVEL_HIGH	Specifies that the trigger is considered valid as long as the level of the source signal is high.
LEVEL_LOW	Specifies that the trigger is considered valid as long as the level of the source signal is low.

PRM.TRG_SOFTWARE¶

Description: Generates an internal trigger. TRG_SOURCE has to be set to 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:

Default value: 0

Usage:

```
xiCam.SetParam(PRM.TRG_SOFTWARE,  val);
```

PRM.TRG_SELECTOR¶

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

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

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

Type: Enumerator.

Default value: FRAME_START

Usage:

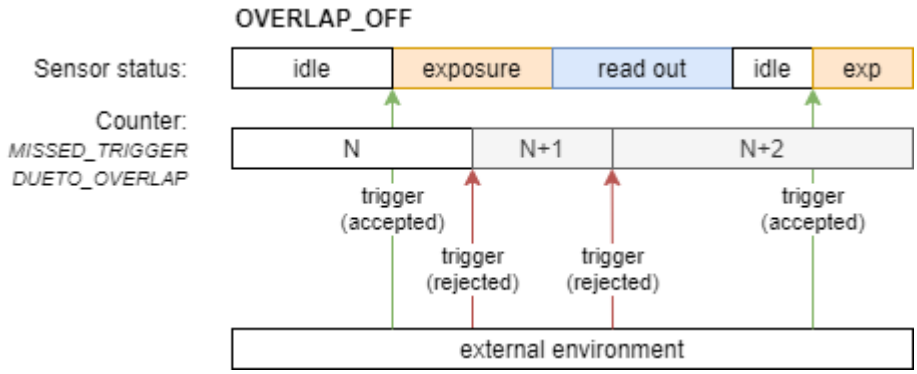
```
xiCam.SetParam(PRM.TRG_SELECTOR, int val);
xiCam.GetParam(PRM.TRG_SELECTOR, out int val);
```

Value	Description
FRAME_START	Trigger starts the capture of one frame
EXPOSURE_ACTIVE	Trigger controls the start and length of the exposure.
FRAME_BURST_START	Trigger starts the capture of the bursts of frames in an acquisition.
FRAME_BURST_ACTIVE	Trigger controls the duration of the capture of the bursts of frames in an acquisition.
MULTIPLE_EXPOSURES	Trigger which when first trigger starts exposure and consequent pulses are gating exposure(active HI)
EXPOSURE_START	Trigger controls the start of the exposure of one Frame.
MULTI_SLOPE_PHASE_CHANGE	Trigger controls the multi slope phase in one Frame (phase0 -> phase1) or (phase1 -> phase2).
ACQUISITION_START	Selects a trigger that starts the Acquisition.

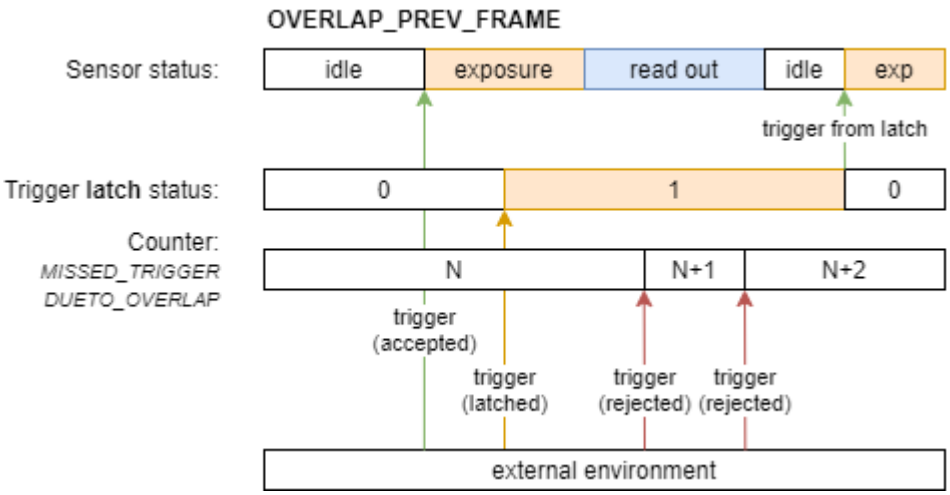
PRM.TRG_OVERLAP¶

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: PREV_FRAME

Is invalidated by: TRG_SELECTOR, EXPOSURE_BURST_COUNT

Usage:

```
xiCam.SetParam(PRM.TRG_OVERLAP, int val);
xiCam.GetParam(PRM.TRG_OVERLAP, out int val);
```

Value	Description
OFF	No trigger overlap is permitted.
READ_OUT	Trigger is accepted immediately after the exposure period. (see Note1)
PREV_FRAME	Trigger is accepted (latched) at any time during the capture of the previous frame.

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

PRM.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:

```
xiCam.SetParam(PRM.ACQ_FRAME_BURST_COUNT, int val);
xiCam.GetParam(PRM.ACQ_FRAME_BURST_COUNT, out int val);
```

PRM.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:

```
xiCam.GetParam(PRM.TIMESTAMP, out ulong val);
```

GPIO Setup¶

PRM.GPI_SELECTOR¶
Description: Selects GPI.

Type: Enumerator.

Default value:
Usage:

```
xiCam.SetParam(PRM.GPI_SELECTOR, int val);
xiCam.GetParam(PRM.GPI_SELECTOR, out int val);
```

Value	Description
GPI_PORT1	GPI port 1
GPI_PORT2	GPI port 2
GPI_PORT3	GPI port 3
GPI_PORT4	GPI port 4
GPI_PORT5	GPI port 5
GPI_PORT6	GPI port 6

GPI_PORT7	GPI port 7
GPI_PORT8	GPI port 8
GPI_PORT9	GPI port 9
GPI_PORT10	GPI port 10
GPI_PORT11	GPI port 11
GPI_PORT12	GPI port 12

PRM.GPI_MODE¶

Description: Defines GPI functionality.

Note1: To use GPI as trigger source, the TRG_SOURCE should be also set to EDGE_RISING or EDGE_FALLING

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

Type: Enumerator.

Default value: OFF

Is invalidated by: GPI_SELECTOR

Usage:

```
xiCam.SetParam(PRM.GPI_MODE, int val);
xiCam.GetParam(PRM.GPI_MODE, out int val);
```

Value	Description
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.
TRIGGER	Input can be used for triggering.
EXT_EVENT	External signal input (not implemented)

PRM.GPI_LEVEL¶

Description: Level of digital input selected by 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 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:

```
xiCam.GetParam(PRM.GPI_LEVEL, out int val);
```

PRM.GPI_LEVEL_AT_IMAGE_EXP_START¶

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

Type: Integer.

Default value: 0

Usage:

```
xiCam.GetParam(PRM.GPI_LEVEL_AT_IMAGE_EXP_START, out int val);
```

PRM.GPI_LEVEL_AT_IMAGE_EXP_END¶

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

Type: Integer.

Default value: 0

Usage:

```
xiCam.GetParam(PRM.GPI_LEVEL_AT_IMAGE_EXP_END, out int val);
```

PRM.GPI_LEVEL_AT_IMAGE_READOUT_END¶

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

Type: Integer.

Default value: 0

Usage:

```
xiCam.GetParam(PRM.GPI_LEVEL_AT_IMAGE_READOUT_END, out int val);
```

PRM.GPO_SELECTOR¶

Description: Selects GPO.

Type: Enumerator.

Default value:

Usage:

```
xiCam.SetParam(PRM.GPO_SELECTOR, int val);
xiCam.GetParam(PRM.GPO_SELECTOR, out int val);
```

Value	Description

GPO_PORT1	GPO port 1
GPO_PORT2	GPO port 2
GPO_PORT3	GPO port 3
GPO_PORT4	GPO port 4
GPO_PORT5	GPO port 5
GPO_PORT6	GPO port 6
GPO_PORT7	GPO port 7
GPO_PORT8	GPO port 8
GPO_PORT9	GPO port 9
GPO_PORT10	GPO port 10
GPO_PORT11	GPO port 11
GPO_PORT12	GPO port 12

PRM.GPO_MODE¶
Description: Defines GPO functionality.

Note1: On some camera models (MR, MH): Modes FRAME_ACTIVE or EXPOSURE_ACTIVE are supported only if TRG_SOURCE is set to SOFTWARE or EDGE_RISING or EDGE_FALLING. See section 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: OFF
Is invalidated by: GPO_SELECTOR
Usage:

```
xiCam.SetParam(PRM.GPO_MODE, int val);
xiCam.GetParam(PRM.GPO_MODE, out int val);
```

Value	Description
OFF	Output is off (zero voltage or switched_off)

ON	Output is on (voltage or switched_on)
FRAME_ACTIVE	Output is on while frame exposure,read,transfer.
FRAME_ACTIVE_NEG	Output is off while frame exposure,read,transfer.
EXPOSURE_ACTIVE	Output is on while frame exposure
EXPOSURE_ACTIVE_NEG	Output is off while frame exposure
FRAME_TRIGGER_READY	Output is on while camera is ready for trigger
FRAME_TRIGGER_READY_NEG	Output is off while camera is ready for trigger.
EXPOSURE_PULSE	Output is on short pulse at the beginning of frame exposure.
EXPOSURE_PULSE_NEG	Output is off short pulse at the beginning of frame exposure.
BUSY	Output is on when camera has received trigger until end of transfer
BUSY_NEG	Output is off when camera has received trigger until end of transfer
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.
FRAME_BUFFER_OVERFLOW	Frame buffer overflow status.
EXPOSURE_ACTIVE_FIRST_ROW	Output is on while the first row exposure.
EXPOSURE_ACTIVE_FIRST_ROW_NEG	Output is off while the first row exposure.
EXPOSURE_ACTIVE_ALL_ROWS	Output is on while all rows exposure together.
EXPOSURE_ACTIVE_ALL_ROWS_NEG	Output is off while all rows exposure together.
TXD	Output is connected to TXD of UART module

PRM.METADATA_SAMPLING_MODE¶

Description: On some cameras (MU) selects metadata sampling point.

Type: Enumerator.

Default value: METADATA_SAMPLING_OFF

Usage:

```
xiCam.SetParam(PRM.METADATA_SAMPLING_MODE, int val);
xiCam.GetParam(PRM.METADATA_SAMPLING_MODE, out int val);
```

Value	Description
METADATA_SAMPLING_OFF	Metadata sampling turned off
METADATA_SAMPLING_POINT_EXPOSURE_START	Selects the sampling point to exposure start
METADATA_SAMPLING_POINT_EXPOSURE_END	Selects the sampling point to exposure end
METADATA_SAMPLING_POINT_EXPOSURE_START_END	Selects the sampling point to exposure start and end
METADATA_SAMPLING_POINT_READOUT_END	Selects the sampling point to readout end

PRM.LED_SELECTOR¶

Description: Selects LED.

Type: Enumerator.

Default value:

Usage:

```
xiCam.SetParam(PRM.LED_SELECTOR, int val);
xiCam.GetParam(PRM.LED_SELECTOR, out int val);
```

Value	Description
LED_SEL1	LED 1
LED_SEL2	LED 2
LED_SEL3	LED 3
LED_SEL4	LED 4
LED_SEL5	LED 5

PRM.LED_MODE¶

Description: Defines LED functionality.

Type: Enumerator.

Default value: LED_HEARTBEAT

Is invalidated by: LED_SELECTOR

Usage:

```
xiCam.SetParam(PRM.LED_MODE, int val);
xiCam.GetParam(PRM.LED_MODE, out int val);
```

Value	Description
LED_HEARTBEAT	Set led to blink (1 Hz) if link is OK.
LED_TRIGGER_ACTIVE	Set led to blink if trigger detected.
LED_EXT_EVENT_ACTIVE	Set led to blink if external signal detected.
LED_LINK	Set led to blink if link is OK.
LED_ACQUISITION	Set led to blink if data streaming
LED_EXPOSURE_ACTIVE	Set led to blink if sensor integration time.
LED_FRAME_ACTIVE	Set led to blink if device busy/not busy.
LED_OFF	Set led to off.
LED_ON	Set led to on.
LED_BLINK	Blinking (1Hz).

PRM.DEBOUNCE_EN¶

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

Note1: Parameter is available only for xiQ camera models.

Type: Integer.

Default value: 0

Usage:

```
xiCam.SetParam(PRM.DEBOUNCE_EN, int val);
xiCam.GetParam(PRM.DEBOUNCE_EN, out int val);
```

Debounce Setup¶

PRM.DEBOUNCE_T0¶

Description: Debounce time (x * 10us) for transition to inactive level of GPI selected by DEBOUNCE_POL .

Type: Integer.

Default value: 0

Usage:

```
xiCam.SetParam(PRM.DEBOUNCE_T0, int val);
xiCam.GetParam(PRM.DEBOUNCE_T0, out int val);
```

PRM.DEBOUNCE_T1¶

Description: Debounce time (x * 10us)for transition to active level of GPI selected by DEBOUNCE_POL

Type: Integer.

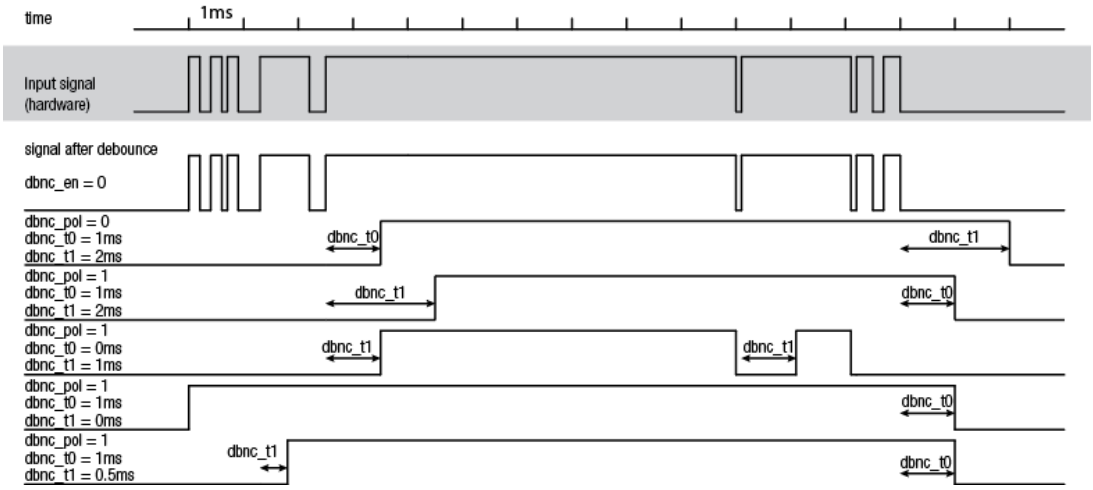
Default value: 0

Usage:

```
xiCam.SetParam(PRM.DEBOUNCE_T1, int val);
xiCam.GetParam(PRM.DEBOUNCE_T1, out int val);
```

PRM.DEBOUNCE_POL¶

Description: Debounce polarity selects active level of GPI (see GPI_SELECTOR parameter). Does not inverts the signal if set.



Type: Integer.

Default value: 0

Typical range: [0, 1]

Usage:

```
xiCam.SetParam(PRM.DEBOUNCE_POL, int val);
xiCam.GetParam(PRM.DEBOUNCE_POL, out int val);
```

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

PRM.LENS_MODE¶

Description: Status of lens control interface. This shall be set to XI_ON before any Lens operations.

Type: Integer.

Default value: 0

Usage:

```
xiCam.SetParam(PRM.LENS_MODE, int val);
xiCam.GetParam(PRM.LENS_MODE, out int val);
```

PRM.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:

```
xiCam.SetParam(PRM.LENS_APERTURE_VALUE, float val);
xiCam.GetParam(PRM.LENS_APERTURE_VALUE, out float val);
```

PRM.LENS_APERTURE_INDEX¶

Description: Current lens aperture motor step value.

Type: Integer.

Default value: 1

Usage:

```
xiCam.SetParam(PRM.LENS_APERTURE_INDEX, int val);
xiCam.GetParam(PRM.LENS_APERTURE_INDEX, out int val);
```

PRM.LENS_FOCUS_MOVEMENT_VALUE¶

Description: Lens current focus movement value to be used by 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:

```
xiCam.SetParam(PRM.LENS_FOCUS_MOVEMENT_VALUE, int val);
xiCam.GetParam(PRM.LENS_FOCUS_MOVEMENT_VALUE, out int val);
```

PRM.LENS_FOCUS_MOVE¶

Description: Moves lens focus motor by steps set in LENS_FOCUS_MOVEMENT_VALUE

Type:

Default value: 0
Usage:

```
xiCam.SetParam(PRM.LENS_FOCUS_MOVE,  val);
```

PRM.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:

```
xiCam.GetParam(PRM.LENS_FOCAL_LENGTH, out float val);
```

PRM.LENS_FEATURE_SELECTOR¶

Description: Selects the current feature which is accessible by LENS_FEATURE

Type: Enumerator.

Default value: MOTORIZED_FOCUS_SWITCH
Usage:

```
xiCam.SetParam(PRM.LENS_FEATURE_SELECTOR, int val);
xiCam.GetParam(PRM.LENS_FEATURE_SELECTOR, out int val);
```

Value	Description
MOTORIZED_FOCUS_SWITCH	Status of lens motorized focus switch

MOTORIZED_FOCUS_BOUNDED	On read = 1 if motorized focus is on one of limits.
MOTORIZED_FOCUS_CALIBRATION	(planned feature) On read = 1 if motorized focus is calibrated. Write 1 to start calibration.
IMAGE_STABILIZATION_ENABLED	On read = 1 if image stabilization is enabled. Write 1 to enable image stabilization.
IMAGE_STABILIZATION_SWITCH_STATUS	On read = 1 if image stabilization switch is in position On.
IMAGE_ZOOM_SUPPORTED	On read = 1 if lens supports zoom = are not prime.

PRM.LENS_FEATURE¶

Description: Allows access to lens feature value currently selected by LENS_FEATURE_SELECTOR .

Type: Float.

Default value: 0.0

Usage:

```
xiCam.SetParam(PRM.LENS_FEATURE, float val);
xiCam.GetParam(PRM.LENS_FEATURE, out float val);
```

Device info parameters¶

PRM.DEVICE_NAME¶

Description: Return device name.

Type: String.

Default value: -

Usage:

```
xiCam.GetParam(PRM.DEVICE_NAME, out string val);
```

PRM.DEVICE_TYPE¶

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

Type: String.

Default value: -

Usage:

```
xiCam.GetParam(PRM.DEVICE_TYPE, out string val);
```

PRM.DEVICE_MODEL_ID¶

Description: Returns the device model id.

Type: Integer.

Default value: 0
Usage:

```
xiCam.GetParam(PRM.DEVICE_MODEL_ID, out int val);
```

PRM.SENSOR_MODEL_ID¶
Description: Returns the device sensor model id.

Type: Integer.

Default value: 0
Usage:

```
xiCam.GetParam(PRM.SENSOR_MODEL_ID, out int val);
```

PRM.DEVICE_SN¶
Description: Returns device serial number. Only string form is possible. It might contain also alphabet characters.

Type: String.

Default value: 0
Usage:

```
xiCam.GetParam(PRM.DEVICE_SN, out string val);
```

PRM.DEVICE_SENS_SN¶
Description: Returns sensor serial number.

Type: String.

Default value: 0
Usage:

```
xiCam.GetParam(PRM.DEVICE_SENS_SN, out string val);
```

PRM.DEVICE_INSTANCE_PATH¶
Description: Returns device instance path in operating system.

Type: String.

Default value: -
Usage:

```
xiCam.GetParam(PRM.DEVICE_INSTANCE_PATH, out string val);
```

PRM.DEVICE_LOCATION_PATH¶
Description: Returns device location path in operating system. It should reflect the connection position.

Type: String.

Default value: -
Usage:

```
xiCam.GetParam(PRM.DEVICE_LOCATION_PATH, out string val);
```

PRM.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:

```
xiCam.GetParam(PRM.DEVICE_USER_ID, out string val);
```

PRM.DEVICE_MANIFEST¶
Description: Get XML of current xiAPI device parameters and capabilities.

Type: String.

Default value: -
Usage:

```
xiCam.GetParam(PRM.DEVICE_MANIFEST, out string val);
```

PRM.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:

```
xiCam.SetParam(PRM.IMAGE_USER_DATA, int val);
```

```
xiCam.GetParam(PRM.IMAGE_USER_DATA, out int val);
```

Device acquisition settings¶

PRM.IMAGE_DATA_FORMAT_RGB32_ALPHA¶

Description: The alpha channel of RGB32 output image format(see IMAGE_DATA_FORMAT).

Type: Integer.

Default value: 0

Typical range: [0, 0xFFFF]

Usage:

```
xiCam.SetParam(PRM.IMAGE_DATA_FORMAT_RGB32_ALPHA, int val);
xiCam.GetParam(PRM.IMAGE_DATA_FORMAT_RGB32_ALPHA, out int val);
```

PRM.IMAGE_PAYLOAD_SIZE¶

Description: Buffer size in bytes sufficient for output image returned by GetImage

Type: Integer.

Default value: 0

Usage:

```
xiCam.GetParam(PRM.IMAGE_PAYLOAD_SIZE, out int val);
```

PRM.TRANSPORT_PIXEL_FORMAT¶

Description: Transport pixel format is format of data transported by link to transport layer. It might be modified after setting of IMAGE_DATA_FORMAT , OUTPUT_DATA_PACKING , OUTPUT_DATA_BIT_DEPTH , ...

Type: Enumerator.

Default value: GenTL_Image_Format_Mono8

Usage:

```
xiCam.SetParam(PRM.TRANSPORT_PIXEL_FORMAT, int val);
xiCam.GetParam(PRM.TRANSPORT_PIXEL_FORMAT, out int val);
```

PRM.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: CPU_RAM

Usage:

```
xiCam.SetParam(PRM.TRANSPORT_DATA_TARGET, int val);
xiCam.GetParam(PRM.TRANSPORT_DATA_TARGET, out int val);
```

Value	Description
CPU_RAM	normal CPU memory buffer is used for image data
GPU_RAM	data is delivered straight to GPU memory using GPUDirect technology
UNIFIED	CUDA managed memory is used for image data.
ZEROCOPY	CUDA zerocopy memory is used for image data.

PRM.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: LIMIT_BANDWIDTH
Usage:

```
xiCam.SetParam(PRM.SENSOR_CLOCK_FREQ_HZ, float val);
xiCam.GetParam(PRM.SENSOR_CLOCK_FREQ_HZ, out float val);
```

PRM.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:

```
xiCam.SetParam(PRM.SENSOR_CLOCK_FREQ_INDEX, int val);
xiCam.GetParam(PRM.SENSOR_CLOCK_FREQ_INDEX, out int val);
```

PRM.SENSOR_OUTPUT_CHANNEL_COUNT¶
Description: Number of output channels from sensor used for data transfer.

Type: Enumerator.

Default value: Depends on sensor model.
Usage:

```
xiCam.SetParam(PRM.SENSOR_OUTPUT_CHANNEL_COUNT, int val);
xiCam.GetParam(PRM.SENSOR_OUTPUT_CHANNEL_COUNT, out int val);
```

Value	Description
CHANN_CNT2	2 sensor readout channels.
CHANN_CNT4	4 sensor readout channels.
CHANN_CNT8	8 sensor readout channels.
CHANN_CNT16	16 sensor readout channels.
CHANN_CNT24	24 sensor readout channels.
CHANN_CNT32	32 sensor readout channels.
CHANN_CNT48	48 sensor readout channels.

PRM.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 MAX , MIN .

Type: Float.

Default value: 0.0

Is invalidated by: IMAGE_DATA_FORMAT, EXPOSURE, ACQ_TIMING_MODE, LIMIT_BANDWIDTH, LIMIT_BANDWIDTH_MODE, DOWNSAMPLING_TYPE, DOWNSAMPLING, BINNING_VERTICAL, BINNING_HORIZONTAL, DECIMATION_VERTICAL, DECIMATION_HORIZONTAL, WIDTH, HEIGHT, OUTPUT_DATA_PACKING, OUTPUT_DATA_PACKING_TYPE, SENSOR_DATA_BIT_DEPTH, OUTPUT_DATA_BIT_DEPTH, DUAL_ADC_MODE, SENSOR_FEATURE_VALUE

Usage:

```
xiCam.SetParam(PRM.FRAMERATE, float val);
xiCam.GetParam(PRM.FRAMERATE, out float val);
```

PRM.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: TRANSPORT_SKIPPED_FRAMES

Usage:

```
xiCam.SetParam(PRM.COUNTER_SELECTOR, int val);
xiCam.GetParam(PRM.COUNTER_SELECTOR, out int val);
```

Value	Description
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.
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.
TRANSPORT_TRANSFERRED_FRAMES	Number of delivered buffers since last acquisition start.
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)
FRAME_MISSED_TRIGGER_DUETO_FRAME_BUFFER_OVR	Number of missed triggers due to frame buffer full. (see Note1)
FRAME_BUFFER_OVERFLOW	Internal camera frame buffer memory (RAM) full events counter. It can be incremented multiple times per one frame. (see Note1)
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)
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).

PRM.COUNTER_VALUE¶

Description: Returns value of selected (by COUNTER_SELECTOR) frame counter.

Note: All counters are reset with the camera open, and counters TRANSPORT_SKIPPED_FRAMES, API_SKIPPED_FRAMES and TRANSPORT_TRANSFERRED_FRAMES are also reset with acquisition start.

Type: Integer.

Default value: 0

Usage:

```
xiCam.GetParam(PRM.COUNTER_VALUE, out int val);
```

PRM.ACQ_TIMING_MODE¶

Description: This parameter defines the acquisition timing mode. More information about enumerators FRAME_RATE and FRAME_RATE_LIMIT please refer to our Frame Rate Control support page.

Type: Enumerator.

Default value: FREE_RUN

Usage:

```
xiCam.SetParam(PRM.ACQ_TIMING_MODE, int val);
xiCam.GetParam(PRM.ACQ_TIMING_MODE, out int val);
```

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

PRM.AVAILABLE_BANDWIDTH¶

Description: Measure available interface bandwidth. Unit is Megabits (1000000) per sec.

Note: Measurement is optimized for default parameters after OpenDevice, which is suggested point for usage. 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:

```
xiCam.GetParam(PRM.AVAILABLE_BANDWIDTH, out int val);
```

PRM.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 IMAGE_PAYLOAD_SIZE

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

Type: Enumerator.

Default value: UNSAFE

Usage:

```
xiCam.SetParam(PRM.BUFFER_POLICY, int val);
xiCam.GetParam(PRM.BUFFER_POLICY, out int val);
```

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

PRM.LUT_EN¶

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 LUT_EN also uploads previously set values in to camera. Values are latched in API.

Type: Integer.

Default value: 0

Usage:

```
xiCam.SetParam(PRM.LUT_EN, int val);
xiCam.GetParam(PRM.LUT_EN, out int val);
```

PRM.LUT_INDEX¶

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 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 MAX , MIN .

Type: Integer.

Default value: 0

Usage:

```
xiCam.SetParam(PRM.LUT_INDEX, int val);
xiCam.GetParam(PRM.LUT_INDEX, out int val);
```

PRM.LUT_VALUE¶

Description: Defines value at entry LUTIndex of the LUT.

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

Note2: All xiQ cameras have LUT N-bit to N-bit, based on the 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 LUT_EN to value 1, will apply all changes to camera.

Type: Integer.

Default value: 0
Usage:

```
xiCam.SetParam(PRM.LUT_VALUE, int val);
xiCam.GetParam(PRM.LUT_VALUE, out int val);
```

PRM.TRG_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: TRG_SELECTOR
Usage:

```
xiCam.SetParam(PRM.TRG_DELAY, int val);
xiCam.GetParam(PRM.TRG_DELAY, out int val);
```

PRM.TS_RST_MODE¶
Description: Defines way timestamp reset engine is armed.

Type: Enumerator.

Default value: ARM_ONCE
Usage:

```
xiCam.SetParam(PRM.TS_RST_MODE, int val);
xiCam.GetParam(PRM.TS_RST_MODE, out int val);
```

Value	Description

ARM_ONCE	Engine is disabled after TimeStamp has been reset after selected event.
ARM_PERSIST	Engine is armed permanently so each selected event will trigger TimeStamp reset.

PRM.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: OFF

Usage:

```
xiCam.SetParam(PRM.TS_RST_SOURCE, int val);
xiCam.GetParam(PRM.TS_RST_SOURCE, out int val);
```

Value	Description
OFF	No source selected TimeStamp reset is not armed.
GPI_1	GPI1 rising edge is active (signal after de-bounce module)
GPI_2	GPI2 rising edge is active
GPI_3	GPI3 rising edge is active
GPI_4	GPI4 rising edge is active
GPI_1_INV	GPI1 falling edge is active
GPI_2_INV	GPI2 falling edge is active
GPI_3_INV	GPI3 falling edge is active
GPI_4_INV	GPI4 falling edge is active
GPO_1	TimeStamp reset source selected GPO1
GPO_2	TimeStamp reset source selected GPO2
GPO_3	TimeStamp reset source selected GPO3
GPO_4	TimeStamp reset source selected GPO4
GPO_1_INV	TimeStamp reset source selected GPO1 inverted
GPO_2_INV	TimeStamp reset source selected GPO2 inverted

GPO_3_INV	TimeStamp reset source selected GPO3 inverted
GPO_4_INV	TimeStamp reset source selected GPO4 inverted
TRIGGER	TRIGGER to sensor rising edge is active
TRIGGER_INV	TRIGGER to sensor rising edge is active
SW	TRIGGER to sensor rising edge is active. TimeStamp is reset by software take effect imminently.
EXPACTIVE	Exposure Active signal rising edge
EXPACTIVE_INV	Exposure Active signal falling edge
FVAL	Frame valid signal rising edge (internal signal in camera)
FVAL_INV	Frame valid signal falling edge (internal signal in camera)
GPI_5	GPI5 rising edge is active
GPI_6	GPI6 rising edge is active
GPI_5_INV	GPI5 falling edge is active
GPI_6_INV	GPI6 falling edge is active
GPI_7	TimeStamp reset source selected GPI7 (after de bounce)
GPI_8	TimeStamp reset source selected GPI8 (after de bounce)
GPI_9	TimeStamp reset source selected GPI9 (after de bounce)
GPI_10	TimeStamp reset source selected GPI10 (after de bounce)
GPI_11	TimeStamp reset source selected GPI11 (after de bounce)
GPI_7_INV	TimeStamp reset source selected GPI7 inverted (after de bounce)
GPI_8_INV	TimeStamp reset source selected GPI8 inverted (after de bounce)
GPI_9_INV	TimeStamp reset source selected GPI9 inverted (after de bounce)
GPI_10_INV	TimeStamp reset source selected GPI10 inverted (after de bounce)
GPI_11_INV	TimeStamp reset source selected GPI11 inverted (after de bounce)

Extended Device parameters¶

PRM.IS_DEVICE_EXIST¶

Description: Returns 1 if camera connected and works properly.

Type: Integer.

Default value: 0
Usage:

```
xiCam.GetParam(PRM.IS_DEVICE_EXIST, out int val);
```

PRM.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:

```
xiCam.SetParam(PRM.ACQ_BUFFER_SIZE, int val);
xiCam.GetParam(PRM.ACQ_BUFFER_SIZE, out int val);
```

PRM.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:

```
xiCam.SetParam(PRM.ACQ_BUFFER_SIZE_UNIT, int val);
xiCam.GetParam(PRM.ACQ_BUFFER_SIZE_UNIT, out int val);
```

PRM.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 (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:

```
xiCam.SetParam(PRM.ACQ_TRANSPORT_BUFFER_SIZE, int val);
xiCam.GetParam(PRM.ACQ_TRANSPORT_BUFFER_SIZE, out int val);
```

PRM.ACQ_TRANSPORT_PACKET_SIZE¶
Description: Acquisition transport packet size in bytes. (only valid for MQ,MD camera families)

Type: Integer.

Default value: 0
Usage:

```
xiCam.SetParam(PRM.ACQ_TRANSPORT_PACKET_SIZE, int val);
xiCam.GetParam(PRM.ACQ_TRANSPORT_PACKET_SIZE, out int val);
```

PRM.BUFFERS_QUEUE_SIZE¶
Description: 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: ACQ_BUFFER_SIZE
Usage:

```
xiCam.SetParam(PRM.BUFFERS_QUEUE_SIZE, int val);
xiCam.GetParam(PRM.BUFFERS_QUEUE_SIZE, out int val);
```

PRM.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:

```
xiCam.SetParam(PRM.ACQ_TRANSPORT_BUFFER_COMMIT, int val);
xiCam.GetParam(PRM.ACQ_TRANSPORT_BUFFER_COMMIT, out int val);
```

PRM.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: 0

Usage:

```
xiCam.SetParam(PRM.RECENT_FRAME, int val);
xiCam.GetParam(PRM.RECENT_FRAME, out int val);
```

PRM.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:

Default value: 0

Usage:

```
xiCam.SetParam(PRM.DEVICE_RESET,  val);
```

PRM.CONCAT_IMG_MODE¶

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

Type: Integer.

Default value: 0

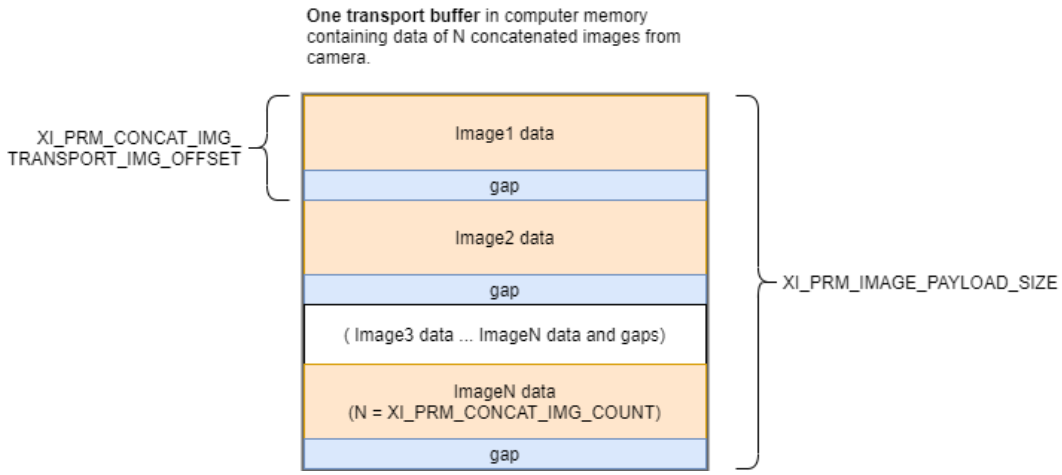
Usage:

```
xiCam.SetParam(PRM.CONCAT_IMG_MODE, int val);
xiCam.GetParam(PRM.CONCAT_IMG_MODE, out int val);
```

PRM.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:

```
xiCam.SetParam(PRM.CONCAT_IMG_COUNT, int val);
xiCam.GetParam(PRM.CONCAT_IMG_COUNT, out int val);
```

PRM.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:

```
xiCam.GetParam(PRM.CONCAT_IMG_TRANSPORT_IMG_OFFSET, out int val);
```

PRM.PROBE_SELECTOR¶
Description: Select Probe

Type: Enumerator.

Default value: PROBE_SELECTOR_CURRENT_MAINBOARD_VCC_IN
Usage:

```
xiCam.SetParam(PRM.PROBE_SELECTOR, int val);
xiCam.GetParam(PRM.PROBE_SELECTOR, out int val);
```

Value	Description

PROBE_SELECTOR_CURRENT_MAINBOARD_VCC_IN	Current probe on Main Board at VCC_IN power supply
PROBE_SELECTOR_VOLTAGE_MAINBOARD_VCC_IN	Voltage probe on Main Board at VCC_IN power supply
PROBE_SELECTOR_CURRENT_MAINBOARD_VCC_ADJ2	Current probe on Main Board at VCC_ADJ2 power supply
PROBE_SELECTOR_VOLTAGE_MAINBOARD_VCC_ADJ2	Voltage probe on Main Board at VCC_ADJ2 power supply
PROBE_SELECTOR_CURRENT_MAINBOARD_VCC_ADJ1	Current probe on Main Board at VCC_ADJ1 power supply
PROBE_SELECTOR_VOLTAGE_MAINBOARD_VCC_ADJ1	Voltage probe on Main Board at VCC_ADJ1 power supply
PROBE_SELECTOR_CURRENT_MAINBOARD_VCC_PLT	Current probe on Main Board at VCC_PLT power supply
PROBE_SELECTOR_VOLTAGE_MAINBOARD_VCC_PLT	Voltage probe on Main Board at VCC_PLT power supply
PROBE_SELECTOR_VOLTAGE_SENSORBOARD_VCC_ADJ1	Voltage probe on Sensor Board at VCC_ADJ1
PROBE_SELECTOR_VOLTAGE_SENSORBOARD_VCC_ADJ2	Voltage probe on Sensor Board at VCC_ADJ2
PROBE_SELECTOR_VOLTAGE_SENSORBOARD_VCC_5V0	Voltage probe on Sensor Board at VCC_5V0
PROBE_SELECTOR_VOLTAGE_SENSORBOARD_VCC_3V3	Voltage probe on Sensor Board at VCC_3V3
PROBE_SELECTOR_VOLTAGE_DATA_CON_INPUT	Voltage probe on device input, if device is bus powered
PROBE_SELECTOR_VOLTAGE_PELTIER1	Voltage probe on peltier #1
PROBE_SELECTOR_CURRENT_PELTIER1	Current probe on peltier #1
PROBE_SELECTOR_VOLTAGE_PELTIER2	Voltage probe on peltier #2
PROBE_SELECTOR_CURRENT_PELTIER2	Current probe on peltier #2
PROBE_SELECTOR_VOLTAGE_MAINBOARD_VCC_5V0	Voltage probe on Main Board at VCC_5V0 power supply
PROBE_SELECTOR_CURRENT_MAINBOARD_VCC_5V0	Current probe on Main Board at VCC_5V0 power supply

PRM.PROBE_VALUE¶

Description: Returns Value of the selected Probe

Type: Float.

Default value: 0.0

Usage:

```
xiCam.GetParam(PRM.PROBE_VALUE, out float val);
```

Sensor Defects Correction¶

PRM.COLUMN_FPN_CORRECTION¶

Description: Correction of column fpn.

Type: Enumerator.

Default value: OFF

Usage:

```
xiCam.SetParam(PRM.COLUMN_FPN_CORRECTION, int val);
xiCam.GetParam(PRM.COLUMN_FPN_CORRECTION, out int val);
```

Value	Description
OFF	Turn parameter off
ON	Turn parameter on

PRM.ROW_FPN_CORRECTION¶

Description: Correction of row fpn.

Type: Enumerator.

Default value: OFF

Usage:

```
xiCam.SetParam(PRM.ROW_FPN_CORRECTION, int val);
xiCam.GetParam(PRM.ROW_FPN_CORRECTION, out int val);
```

Value	Description
OFF	Turn parameter off
ON	Turn parameter on

PRM.COLUMN_BLACK_OFFSET_CORRECTION¶

Description: Correction of column black offset.

Type: Enumerator.

Default value: OFF

Usage:

```
xiCam.SetParam(PRM.COLUMN_BLACK_OFFSET_CORRECTION, int val);
xiCam.GetParam(PRM.COLUMN_BLACK_OFFSET_CORRECTION, out int val);
```

Value	Description
OFF	Turn parameter off
ON	Turn parameter on

PRM.ROW_BLACK_OFFSET_CORRECTION¶

Description: Correction of row black offset.

Type: Enumerator.

Default value: OFF

Usage:

```
xiCam.SetParam(PRM.ROW_BLACK_OFFSET_CORRECTION, int val);
xiCam.GetParam(PRM.ROW_BLACK_OFFSET_CORRECTION, out int val);
```

Value	Description
OFF	Turn parameter off
ON	Turn parameter on

Sensor features¶

PRM.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:

Usage:

```
xiCam.SetParam(PRM.SENSOR_MODE, int val);
xiCam.GetParam(PRM.SENSOR_MODE, out int val);
```

Value	Description
SENS_MD0	Sensor mode number 0
SENS_MD1	Sensor mode number 1

SENS_MD2	Sensor mode number 2
SENS_MD3	Sensor mode number 3
SENS_MD4	Sensor mode number 4
SENS_MD5	Sensor mode number 5
SENS_MD6	Sensor mode number 6
SENS_MD7	Sensor mode number 7
SENS_MD8	Sensor mode number 8
SENS_MD9	Sensor mode number 9
SENS_MD10	Sensor mode number 10
SENS_MD11	Sensor mode number 11
SENS_MD12	Sensor mode number 12
SENS_MD13	Sensor mode number 13
SENS_MD14	Sensor mode number 14
SENS_MD15	Sensor mode number 15

PRM.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: 0

Usage:

```
xiCam.SetParam(PRM.HDR, int val);
xiCam.GetParam(PRM.HDR, out int val);
```

PRM.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:

```
xiCam.SetParam(PRM.HDR_KNEEPOINT_COUNT, int val);
xiCam.GetParam(PRM.HDR_KNEEPOINT_COUNT, out int val);
```

PRM.HDR_T1¶
Description: Exposure time (T1) of 1st kneepoint in % of EXPOSURE

Type: Integer.

Default value: 60
Typical range: [0, 100]
Usage:

```
xiCam.SetParam(PRM.HDR_T1, int val);
xiCam.GetParam(PRM.HDR_T1, out int val);
```

PRM.HDR_T2¶
Description: Exposure time (T2) of 2nd kneepoint in % of

Type: Integer.

Default value: 80
Typical range: [0, 100]
Usage:

```
xiCam.SetParam(PRM.HDR_T2, int val);
xiCam.GetParam(PRM.HDR_T2, out int val);
```

PRM.KNEEPOINT1¶
Description: Saturation level (SL1) of 1st kneepoint in % of sensor saturation.

Type: Integer.

Default value: 40
Typical range: [0, 100]
Usage:

```
xiCam.SetParam(PRM.KNEEPOINT1, int val);
xiCam.GetParam(PRM.KNEEPOINT1, out int val);
```

PRM.KNEEPOINT2¶
Description: Saturation level (SL2) of 2nd kneepoint in % of sensor saturation.

Type: Integer.

Default value: 60

Typical range: [0, 100]
Usage:

```
xiCam.SetParam(PRM.KNEEPOINT2, int val);
xiCam.GetParam(PRM.KNEEPOINT2, out int val);
```

PRM.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:

```
xiCam.GetParam(PRM.IMAGE_BLACK_LEVEL, out int val);
```

PRM.IMAGE_AREA¶
Description: Defines image area of sensor as output.

Type: Enumerator.

Default value: IMAGE_AREA_ACTIVE
Usage:

```
xiCam.SetParam(PRM.IMAGE_AREA, int val);
xiCam.GetParam(PRM.IMAGE_AREA, out int val);
```

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

PRM.DUAL_ADC_MODE¶
Description: Sets DualADC Mode

Type: Enumerator.

Default value: DUAL_ADC_MODE_OFF
Usage:

```
xiCam.SetParam(PRM.DUAL_ADC_MODE, int val);
xiCam.GetParam(PRM.DUAL_ADC_MODE, out int val);
```

Value	Description

DUAL_ADC_MODE_OFF	Disable DualADC feature
DUAL_ADC_MODE_COMBINED	Set Combined mode
DUAL_ADC_MODE_NON_COMBINED	Set NonCombined mode

PRM.DUAL_ADC_GAIN_RATIO¶

Description: Sets DualADC Gain Ratio in dB

Type: Float.

Default value: 0.0
Typical range: [0.0, 24.0]
Usage:

```
xiCam.SetParam(PRM.DUAL_ADC_GAIN_RATIO, float val);
xiCam.GetParam(PRM.DUAL_ADC_GAIN_RATIO, out float val);
```

PRM.DUAL_ADC_THRESHOLD¶

Description: Sets DualADC Threshold value

Type: Integer.

Default value: 50
Typical range: [0, 100]
Is invalidated by: DUAL_ADC_MODE
Usage:

```
xiCam.SetParam(PRM.DUAL_ADC_THRESHOLD, int val);
xiCam.GetParam(PRM.DUAL_ADC_THRESHOLD, out int val);
```

PRM.COMPRESSION_REGION_SELECTOR¶

Description: Sets Compression Region Selector

Type: Integer.

Default value: 1
Typical range: [1, 2]
Is invalidated by: DUAL_ADC_MODE
Usage:

```
xiCam.SetParam(PRM.COMPRESSION_REGION_SELECTOR, int val);
xiCam.GetParam(PRM.COMPRESSION_REGION_SELECTOR, out int val);
```

PRM.COMPRESSION_REGION_START¶

Description: Sets Compression Region Start

Type: Float.

Default value: 0.0

Typical range: [0.0, 50.0]

Is invalidated by: DUAL_ADC_MODE, DUAL_ADC_GAIN_RATIO, COMPRESSION_REGION_SELECTOR

Usage:

```
xiCam.SetParam(PRM.COMPRESSION_REGION_START, float val);
xiCam.GetParam(PRM.COMPRESSION_REGION_START, out float val);
```

PRM.COMPRESSION_REGION_GAIN¶

Description: Sets Compression Region Gain

Type: Float.

Default value: 0.0

Typical range: [-90.0, 0.0]

Is invalidated by: DUAL_ADC_MODE

Usage:

```
xiCam.SetParam(PRM.COMPRESSION_REGION_GAIN, float val);
xiCam.GetParam(PRM.COMPRESSION_REGION_GAIN, out float val);
```

Version info¶

PRM.VERSION_SELECTOR¶

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

Type: Enumerator.

Default value:

Usage:

```
xiCam.SetParam(PRM.VERSION_SELECTOR, int val);
xiCam.GetParam(PRM.VERSION_SELECTOR, out int val);
```

Value	Description
VER_API	version of API
VER_DRV	version of device driver
VER_MCU1	version of MCU1 firmware.
VER_MCU2	version of MCU2 firmware.
VER_MCU3	version of MCU3 firmware.
VER_FPGA1	version of FPGA1 firmware.

VER_XMLMAN	version of XML manifest.
VER_HW_REV	version of hardware revision.
VER_FACTORY_SET	version of factory set.

PRM.VERSION¶

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

Type: String.

Default value: -

Usage:

```
xiCam.GetParam(PRM.VERSION, out string val);
```

PRM.API_VERSION¶

Description: Returns the version of API.

Type: String.

Default value: -

Usage:

```
xiCam.GetParam(PRM.API_VERSION, out string val);
```

PRM.DRV_VERSION¶

Description: Returns the version of the current device driver.

Type: String.

Default value: -

Usage:

```
xiCam.GetParam(PRM.DRV_VERSION, out string val);
```

PRM.MCU1_VERSION¶

Description: Returns the version of the current MCU1 firmware.

Type: String.

Default value: -

Usage:

```
xiCam.GetParam(PRM.MCU1_VERSION, out string val);
```

PRM.MCU2_VERSION¶

Description: Returns the version of the current MCU2 firmware.

Type: String.

Default value: -
Usage:

```
xiCam.GetParam(PRM.MCU2_VERSION, out string val);
```

PRM.MCU3_VERSION¶
Description: Returns the version of the current MCU3 firmware.

Type: String.

Default value: -
Usage:

```
xiCam.GetParam(PRM.MCU3_VERSION, out string val);
```

PRM.FPGA1_VERSION¶
Description: Returns version of FPGA firmware currently running.

Type: String.

Default value: -
Usage:

```
xiCam.GetParam(PRM.FPGA1_VERSION, out string val);
```

PRM.XMLMAN_VERSION¶
Description: Returns version of XML manifest.

Type: String.

Default value: -
Usage:

```
xiCam.GetParam(PRM.XMLMAN_VERSION, out string val);
```

PRM.HW_REVISION¶
Description: Returns the hardware revision number of the camera.

Type: String.

Default value: -
Usage:

```
xiCam.GetParam(PRM.HW_REVISION, out string val);
```

PRM.FACTORY_SET_VERSION¶
Description: Returns version of factory set.

Type: String.

Default value: -

Usage:

```
xiCam.GetParam(PRM.FACTORY_SET_VERSION, out string val);
```

API features¶

PRM.DEBUG_LEVEL¶

Description: Setting the API debug level allows to select amount of messages stored to debug output. This should be done as the first API call.

Type: Enumerator.

Default value: WARNING

Usage:

```
xiCam.SetParam(PRM.DEBUG_LEVEL, int val);
xiCam.GetParam(PRM.DEBUG_LEVEL, out int val);
```

Value	Description
DETAIL	(see Note1)
TRACE	Prints errors, warnings and important informations
WARNING	Prints all errors and warnings
ERROR	Prints all errors
FATAL	Prints only important errors
DISABLED	Prints no messages

Note1: Prints same as TRACE plus locking of resources.

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

Note3: In Linux the messages are printed to stderr

PRM.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: 1
Usage:

```
xiCam.SetParam(PRM.AUTO_BANDWIDTH_CALCULATION, int val);
xiCam.GetParam(PRM.AUTO_BANDWIDTH_CALCULATION, out int val);
```

PRM.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: 1
Usage:

```
xiCam.SetParam(PRM.NEW_PROCESS_CHAIN_ENABLE, int val);
xiCam.GetParam(PRM.NEW_PROCESS_CHAIN_ENABLE, out int val);
```

PRM.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:

```
xiCam.SetParam(PRM.PROC_NUM_THREADS, int val);
xiCam.GetParam(PRM.PROC_NUM_THREADS, out int val);
```

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 using xiAPI.NET.

PRM.READ_FILE_FFS¶

Description: File data to be read from camera flash file system.

Type: String.

Default value: -

Usage:

```
xiCam.GetParam(PRM.READ_FILE_FFS, out string val);
xiCam.GetParam(PRM.READ_FILE_FFS, out byte[] val, out int val_len);
```

PRM.WRITE_FILE_FFS¶

Description: File data to be written to camera flash file system.

Type: String.

Default value: -

Usage:

```
xiCam.SetParam(PRM.WRITE_FILE_FFS, string val);
```

PRM.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:

```
xiCam.SetParam(PRM.FFS_FILE_NAME, string val);
xiCam.GetParam(PRM.FFS_FILE_NAME, out string val);
```

PRM.FFS_FILE_ID¶

Description: File number(id) in camera FFS.

Type: Integer.

Default value: 0

Usage:

```
xiCam.GetParam(PRM.FFS_FILE_ID, out int val);
```

PRM.FFS_FILE_SIZE¶

Description: Size of a file specified with parameter FFS_FILE_ID in bytes.

Type: Integer.

Default value: 0

Usage:

```
xiCam.GetParam(PRM.FFS_FILE_SIZE, out int val);
```

PRM.FREE_FFS_SIZE¶

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

Type: Unsigned integer 64 bit.

Default value: 0

Usage:

```
xiCam.GetParam(PRM.FREE_FFS_SIZE, out ulong val);
```

PRM.USED_FFS_SIZE¶

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

Type: Unsigned integer 64 bit.

Default value: 0

Usage:

```
xiCam.GetParam(PRM.USED_FFS_SIZE, out ulong val);
```

PRM.FFS_ACCESS_KEY¶

Description: Setting of the key enables file operations on some cameras. It is required to set before usage of WRITE_FILE_FFS .

Type: Integer.

Default value: 0

Usage:

```
xiCam.SetParam(PRM.FFS_ACCESS_KEY, int val);
xiCam.GetParam(PRM.FFS_ACCESS_KEY, out int val);
```

APIContextControl¶

PRM.API_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:

```
xiCam.SetParam(PRM.API_CONTEXT_LIST, string val);
xiCam.GetParam(PRM.API_CONTEXT_LIST, out string val);
```

Sensor Control¶

Note: Some of XIMEA cameras have sensors with specific features.

PRM.SENSOR_FEATURE_SELECTOR¶

Description: Selects the current feature which is accessible by SENSOR_FEATURE_VALUE . See more at our support page, SENSOR FEATURE SELECTOR.

Type: Enumerator.

Default value: SENSOR_FEATURE_ZEROROT_ENABLE

Usage:

```
xiCam.SetParam(PRM.SENSOR_FEATURE_SELECTOR, int val);
xiCam.GetParam(PRM.SENSOR_FEATURE_SELECTOR, out int val);
```

Value	Description
SENSOR_FEATURE_ZEROROT_ENABLE	Sensor Zero ROT enable for ONSEMI PYTHON family. For camera model:MQ013xG-ON (on/off)
SENSOR_FEATURE_BLACK_LEVEL_CLAMP	Black level offset clamping (value). for Camera model:MD
SENSOR_FEATURE_MD_FPGA_DIGITAL_GAIN_DISABLE	Disable digital component of gain for MD family (1=disabled/0=enabled)
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
SENSOR_FEATURE_TIMING_MODE	Sensor timing mode (value depends on sensor)
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
SENSOR_FEATURE_BLACK_LEVEL_OFFSET_RAW	Sensor specific register raw black level offset (value)

SENSOR_FEATURE_SHORT_INTERVAL_SHUTTER	Sensor short Interval Shutter (on/off)
SENSOR_FEATURE_AUTO_LOW_POWER_MODE_AUTO	Sensor low power mode (on/off)
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.
SENSOR_FEATURE_DUAL_TRG_EXP_ZONE_DIVIDER_POSITION	Sensor Dual Trigger Exposure Zone Divider Position
SENSOR_FEATURE_TOF_VCSEL_CTRL_VOLTAGE_MV	ToF VCSEL Control Voltage in mV
SENSOR_FEATURE_MULTIPLE_ADC	Multiple ADC feature performs multiple ADC conversions of the same analog pixel signal and averages the results. It reduces random noise and improves the overall signal-to-noise ratio. For camera model: MX2457 (1=Off, 2=2ADC, 4=4ADC)
SENSOR_FEATURE_FAST_TRIGGER_MODE_ENABLE	Enables a sensor trigger mode that starts exposure immediately at the beginning of the trigger signal. When fast trigger mode is disabled, the camera operates in a trigger mode that allows exposure and readout to overlap, resulting in a higher FPS. For camera model: MU051xG. (on/off)

PRM.SENSOR_FEATURE_VALUE¶

Description: Allows access to sensor feature value currently selected by SENSOR_FEATURE_SELECTOR

Type: Integer.

Default value: 0

Typical range: [0, 1024]

Is invalidated by: SENSOR_FEATURE_SELECTOR

Usage:

```
xiCam.SetParam(PRM.SENSOR_FEATURE_VALUE, int val);
```

```
xiCam.GetParam(PRM.SENSOR_FEATURE_VALUE, out int val);
```

Extended Features¶

PRM.ACQUISITION_STATUS_SELECTOR¶

Description: Selects the internal acquisition signal to read using ACQUISITION_STATUS

Type: Enumerator.

Default value: ACQUISITION_STATUS_ACQ_ACTIVE

Usage:

```
xiCam.SetParam(PRM.ACQUISITION_STATUS_SELECTOR, int val);
xiCam.GetParam(PRM.ACQUISITION_STATUS_SELECTOR, out int val);
```

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

PRM.ACQUISITION_STATUS¶

Description: Returns status of acquisition.

Type: Enumerator.

Default value: OFF

Usage:

```
xiCam.GetParam(PRM.ACQUISITION_STATUS, out int val);
```

Value	Description
OFF	Turn parameter off
ON	Turn parameter on

PRM.DP_UNIT_SELECTOR¶

Description: Data Pipe Unit Selector.

Type: Enumerator.

Default value: SENSOR

Usage:

```
xiCam.SetParam(PRM.DP_UNIT_SELECTOR, int val);
xiCam.GetParam(PRM.DP_UNIT_SELECTOR, out int val);
```

Value	Description

SENSOR	Selects device image sensor
FPGA	Selects device image FPGA

PRM.DP_PROC_SELECTOR¶

Description: Data Pipe Processor Selector.

Type: Enumerator.

Default value: NONE

Is invalidated by: DP_UNIT_SELECTOR

Usage:

```
xiCam.SetParam(PRM.DP_PROC_SELECTOR, int val);
xiCam.GetParam(PRM.DP_PROC_SELECTOR, out int val);
```

Value	Description
NONE	Default empty processor
CHANNEL_MUXER	Channel Muxer (selected processor combines multiple input channels)
PIXEL_SEQUENCER	Selects pixel data output sequence
CHANNEL_1	Selects sensor output channel 1
CHANNEL_2	Selects sensor output channel 2
FRAME_BUFFER	Selects frame buffer memory

PRM.DP_PARAM_SELECTOR¶

Description: Data Pipe Processor parameter Selector.

Type: Enumerator.

Default value: NONE

Is invalidated by: DP_UNIT_SELECTOR, DP_PROC_SELECTOR

Usage:

```
xiCam.SetParam(PRM.DP_PARAM_SELECTOR, int val);
xiCam.GetParam(PRM.DP_PARAM_SELECTOR, out int val);
```

Value	Description
NONE	Empty parameter
CHMUX_CHANNEL_SELECTOR	Defines output of Channel Muxer processor
CHMUX_ALPHA	Channel merger coefficient Alpha

CHMUX_BETA	Channel merger coefficient Beta
PIXSEQ_SELECTOR	PixSeq Selector
CHANNEL_TIMING	Selected channel timing
FRAMEBUF_MODE	Frame Buffer Mode
FRAMEBUF_SIZE	Frame Buffer Size Bytes

PRM.DP_PARAM_VALUE¶

Description: Data Pipe processor parameter value.

Type: Float.

Default value: 0.0
Typical range: [0.0, 100000.0]
Is invalidated by: DP_UNIT_SELECTOR, DP_PROC_SELECTOR, DP_PARAM_SELECTOR
Usage:

```
xiCam.SetParam(PRM.DP_PARAM_VALUE, float val);
xiCam.GetParam(PRM.DP_PARAM_VALUE, out float val);
```

PRM.GENTL_DATASTREAM_ENABLED¶

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: 0
Usage:

```
xiCam.SetParam(PRM.GENTL_DATASTREAM_ENABLED, int val);
xiCam.GetParam(PRM.GENTL_DATASTREAM_ENABLED, out int val);
```

PRM.GENTL_DATASTREAM_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:

```
xiCam.GetParam(PRM.GENTL_DATASTREAM_CONTEXT, out string val);
```

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.

PRM.USER_SET_SELECTOR¶

Description: User Set to be loaded by USER_SET_LOAD .

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

Type: Enumerator.

Default value:

Usage:

```
xiCam.SetParam(PRM.USER_SET_SELECTOR, int val);
xiCam.GetParam(PRM.USER_SET_SELECTOR, out int val);
```

Value	Description
US_12_STD_L	12bit per channel STD Low Gain mode preset.
US_12_STD_H	12bit per channel STD High Gain mode preset.
US_14_STD_L	14bit per channel STD Low Gain mode preset.
US_NONE	No preset selected.
US_14_STD_H	14bit per channel STD High Gain mode preset.
US_2_12_CMS_S_L	12bit per channel, 2 samples, CMS(summing) Low Gain mode preset.
US_2_12_CMS_S_H	12bit per channel, 2 samples, CMS(summing) High Gain mode preset.
US_2_14_CMS_S_L	14bit per channel, 2 samples, CMS(summing) Low Gain mode preset.
US_2_14_CMS_S_H	14bit per channel, 2 samples, CMS(summing) High Gain mode preset.
US_4_12_CMS_S_L	12bit per channel, 4 samples, CMS(summing) Low Gain mode preset.
US_4_12_CMS_S_H	12bit per channel, 4 samples, CMS(summing) High Gain mode preset.
US_4_14_CMS_S_L	14bit per channel, 4 samples, CMS(summing) Low Gain mode preset.
US_4_14_CMS_S_H	14bit per channel, 4 samples, CMS(summing) High Gain mode preset.

US_2_12_HDR_HL	12bit per channel, 2 samples, HDR High Low Gain mode preset.
US_2_12_HDR_L	12bit per channel, 2 samples, HDR Low Gain mode preset.
US_2_12_HDR_H	12bit per channel, 2 samples, HDR High Gain mode preset.
US_4_12_CMS_HDR_HL	12bit per channel, 4 samples, CMS + HDR High Low Gain mode preset.
US_2_14_HDR_L	14bit per channel, 2 samples, HDR Low Gain mode preset.
US_2_14_HDR_H	14bit per channel, 2 samples, HDR High Gain mode preset.
US_2_12_CMS_A_L	12bit per channel, 2 samples, CMS(averaging) Low Gain mode preset.
US_2_12_CMS_A_H	12bit per channel, 2 samples, CMS(averaging) High Gain mode preset.
US_TOF_DEFAULT	
US_TOF_LONG_RANGE	
US_TOF_SINGLE_READOUT	
US_TOF_DEFAULT_EXT_CDS	
US_TOF_LONG_RANGE_EXT_CDS	
US_TOF_SINGLE_READOUT_EXT_CDS	

PRM.USER_SET_LOAD¶

Description: Loads User Set selected by 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:

Default value: 0

Usage:

```
xiCam.SetParam(PRM.USER_SET_LOAD,  val);
```

PRM.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:

Usage:

```
xiCam.SetParam(PRM.USER_SET_DEFAULT, int val);
xiCam.GetParam(PRM.USER_SET_DEFAULT, out int val);
```

Value	Description
US_12_STD_L	12bit per channel STD Low Gain mode preset.
US_12_STD_H	12bit per channel STD High Gain mode preset.
US_14_STD_L	14bit per channel STD Low Gain mode preset.
US_NONE	No preset selected.
US_14_STD_H	14bit per channel STD High Gain mode preset.
US_2_12_CMS_S_L	12bit per channel, 2 samples, CMS(summing) Low Gain mode preset.
US_2_12_CMS_S_H	12bit per channel, 2 samples, CMS(summing) High Gain mode preset.
US_2_14_CMS_S_L	14bit per channel, 2 samples, CMS(summing) Low Gain mode preset.
US_2_14_CMS_S_H	14bit per channel, 2 samples, CMS(summing) High Gain mode preset.
US_4_12_CMS_S_L	12bit per channel, 4 samples, CMS(summing) Low Gain mode preset.
US_4_12_CMS_S_H	12bit per channel, 4 samples, CMS(summing) High Gain mode preset.
US_4_14_CMS_S_L	14bit per channel, 4 samples, CMS(summing) Low Gain mode preset.
US_4_14_CMS_S_H	14bit per channel, 4 samples, CMS(summing) High Gain mode preset.
US_2_12_HDR_HL	12bit per channel, 2 samples, HDR High Low Gain mode preset.
US_2_12_HDR_L	12bit per channel, 2 samples, HDR Low Gain mode preset.

US_2_12_HDR_H	12bit per channel, 2 samples, HDR High Gain mode preset.
US_4_12_CMS_HDR_HL	12bit per channel, 4 samples, CMS + HDR High Low Gain mode preset.
US_2_14_HDR_L	14bit per channel, 2 samples, HDR Low Gain mode preset.
US_2_14_HDR_H	14bit per channel, 2 samples, HDR High Gain mode preset.
US_2_12_CMS_A_L	12bit per channel, 2 samples, CMS(averaging) Low Gain mode preset.
US_2_12_CMS_A_H	12bit per channel, 2 samples, CMS(averaging) High Gain mode preset.
US_TOF_DEFAULT	
US_TOF_LONG_RANGE	
US_TOF_SINGLE_READOUT	
US_TOF_DEFAULT_EXT_CDS	
US_TOF_LONG_RANGE_EXT_CDS	
US_TOF_SINGLE_READOUT_EXT_CDS	

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).

PRMM.SETTABLE¶

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

Usage:

```
myCam.SetParam(PRM.TEMP_SELECTOR + PRMM.SETTABLE,
TEMP_SELECTOR.TEMP_SENSOR_BOARD);
// if not settable - exception is thrown
```

PRMM.MIN¶

Description: Acquire parameter minimum value

Usage:

```
xiCam.GetParam(PRM.EXPOSURE + PRMM.MIN, out int exp_min);
```

PRMM.MAX¶

Description: Acquire parameter maximum value.

Usage:

```
xiCam.GetParam(PRM.EXPOSURE + PRMM.MAX, out int exp_max);
```

PRMM.INCREMENT¶

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

Usage:

```
xiCam.GetParam(PRM.EXPOSURE + PRMM.INCREMENT, out int exp_inc);
```

PRMM.REQ_VAL_BUFFER_SIZE¶

Description: Parameter modifier for getting required value buffer size for GetParam (e.g. PRM.DEVICE_MANIFEST + PRMM.REQ_BUFFER_SIZE can be used to get needed size of buffer for manifest).

Usage:

```
xiCam.GetParam(PRM.EXPOSURE + PRMM.REQ_VAL_BUFFER_SIZE, out int
exp_req_buf_size);
```

PRMM.DIRECT_UPDATE¶

Description: Parameter modifier for direct update without stopping the streaming. Currently EXPOSURE and GAIN can be used with this modifier.

Usage:

```
xiCam.GetParam(PRM.DEVICE_MANIFEST + PRMM.REQ_VAL_BUFFER_SIZE, out
int manifest_size);
```

Generated:5900c48: Mon Jan 26 03:05:03 CET 2026