



Mobile Robot Vision Expert

Camera SDK (C/C++) Development Guide

April 15, 2024

Contents

1 Overview	5
2 Environment Setup	5
2.1 System configurations for host computer.....	5
2.2 Environment Setup	6
2.2.1 Hardware setup.....	6
2.2.2 Network setup	6
2.2.3 Software environment setup	6
3 Camera SDK Instructions.....	7
3.1 Camera SDK directory	7
3.2 Development process	8
3.2.1 Project setup	8
3.2.2 Call APIs synchronously.....	9
3.2.3 Call APIs asynchronously.....	10
4 Precautions for calling the Camera SDK.....	11
4.1 Principles for calling the Camera SDK	11
4.2 Prerequisites for calling the Camera SDK interfaces.....	11
5 Data type specification.....	13
5.1 LX_STATE	13
5.2 LX_DATA_TYPE.....	15
5.3 LX_BINNING_MODE	15
5.4 LX_STRUCT_LIGHT_CODE_MODE	17
5.5 LX_ALGORITHM_MODE	17
5.6 LX_CAMERA_WORK_MODE.....	17
5.7 LX_TRIGGER_MODE	18
5.8 LX_CAMERA_FEATURE	18
5.8.1 INT type parameters	18
5.8.2 Float type parameters.....	22
5.8.3 Bool type parameters.....	24

5.8.4 String type parameters	25
5.8.5 Command type parameters	25
5.8.6 Ptr type parameters	27
5.9 LxDeviceInfo	29
5.10 LxIntValueInfo	30
5.11 LxFloatValueInfo	30
5.12 LX_DEVICE_TYPE	31
5.13 LX_OPEN_MODE	31
5.14 FrameDataInfo	32
5.15 FrameInfo	32
6 API specification	34
6.1 Find, connect, and turn off the camera	34
6.1.1 DcGetDeviceList	34
6.1.2 DcOpenDevice	34
6.1.3 DcCloseDevice	35
6.2 Start and stop data stream	36
6.2.1 DcStartStream	36
6.2.2 DcStopStream	36
6.3 Get and set the camera parameters	37
6.3.1 DcSetIntValue	37
6.3.2 DcGetIntValue	37
6.3.3 DcSetFloatValue	38
6.3.4 DcGetFloatValue	38
6.3.5 DcSetBoolValue	38
6.3.6 DcGetBoolValue	39
6.3.7 DcSetStringValue	39
6.3.8 DcGetStringValue	40
6.3.9 DcGetPtrValue	40
6.3.10 DcSetCmd	41
6.4 Save point cloud	41

6.4.1 DcSaveXYZ	41
6.5 Special control	42
6.5.1 DcSpecialControl	42
6.6 Set the ROI and parameter paths	42
6.6.1 DcSetRol	42
6.7 Set camera IP	43
6.7.1 DcSetCameraIp	43
6.8 Log and version information	44
6.8.1 DcGetApiVersion	44
6.8.2 DcSetInfoOutput	44
6.8.3 DcLog	45
6.8.4 DcRegisterFrameCallback	45
6.8.5 DcUnregisterFrameCallback	46
6.9 Device status callback	46
6.9.1 DcRegisterCameraStatusCallback	46
6.9.2 DcUnregisterCameraStatusCallback	46
7 Versions and updates	47
7.1 The latest versions and updates	47

1 Overview

This document is mainly for developers related to camera integration and provides instructions on how to use the APIs provided by the Camera SDK which can run on x86 Windows and x86/ARM Linux. This document can be used for M4, M4 Lite, M4 Mega, M4 Pro, V1 Pro, S2, S2 Lite, S2 Max, and H3310 products.

The Camera SDK provides a unified interface for access control and parameter configuration, and supports acquisition of data (including depth images, intensity images, point clouds, RGB images) synchronously or asynchronously. Some camera models support obstacle avoidance, docking, positioning and other data.

This document is intended to assist in the development to the fullest extent possible, and the content provided may be flawed or incomplete.

2 Environment Setup

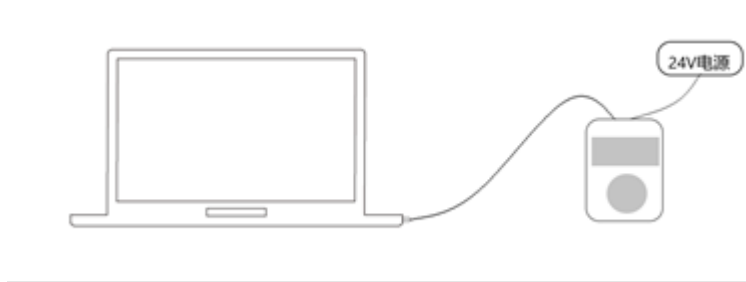
2.1 System configurations for host computer

Items	Configurations
OS	Windows7 Windows10 Windows11 Ubuntu16.04 or above
Memory	4G or above
CPU	Quad-core Cortex-A57 or equivalent or above
Network card	Gigabit or above

2.2 Environment Setup

2.2.1 Hardware setup

Connect the camera to an external voltage of 24V, and to the host computer as shown in the figure above.



2.2.2 Network setup

The default IP range of the camera is: 192.168.100.*.

On the host computer, set the IP address of the network port connected to the camera to an address in the network segment of 192.168.100.x and ensure that it does not conflict with the device, and the subnet mask is 255.255.255.0.

When connecting to a camera, it is suggested to disable the firewall of the host computer to prevent camera packets from being intercepted.

The network configuration of Linux system is the same as that of Windows, and the method of disabling the firewall may be different in different systems, take ubuntu as an example:

Stop the firewall service: `sudo systemctl stop ufw.service`

Disable firewall service: `sudo systemctl disable ufw.service`

Check whether the firewall service is disabled: `sudo ufw status`

2.2.3 Software environment setup

Windows installation:

Open the installation package, install Lanxin-MRDVS-xxx.exe. To install it for all users, administrator privileges are needed to run the installation package.

Linux installation:

Unzip and execute the installation script

```
tar xvf Lanxin-MRDVS-xxx.tar.gz
```

```
cd Lanxin-MRDVS
```

Switch to root user if current user has no root privileges

```
sudo su root
```

And then start the installation

```
chmod +x install.sh
```

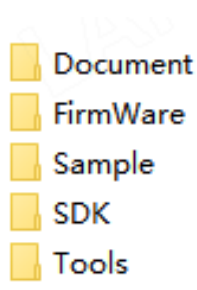
```
./install.sh
```

The installation script installs the SDK to the /opt/lanxin-MRDVS directory and sets the environment variables. If the runner is not the root user, you need to re-run the install.sh script to set the environment variables or manually add `export LD_LIBRARY_PATH=/opt/Lanxin-MRDVS/lib/:$LD_LIBRARY_PATH` to the end of the current user's ~/.bashrc. For terminals that have been opened before installation, it is necessary to update the environment variable of LD_LIBRARY_PATH by running `source ~/.bashrc` before the program runs.

3 Camera SDK Instructions

3.1 Camera SDK directory

The contents of the Camera SDK directory are shown in the figure below:



Document: contains various documentation for the Camera SDK

Firmware: contains firmware upgrade packages

Sample: contains sample code for various programming languages using the Camera SDK

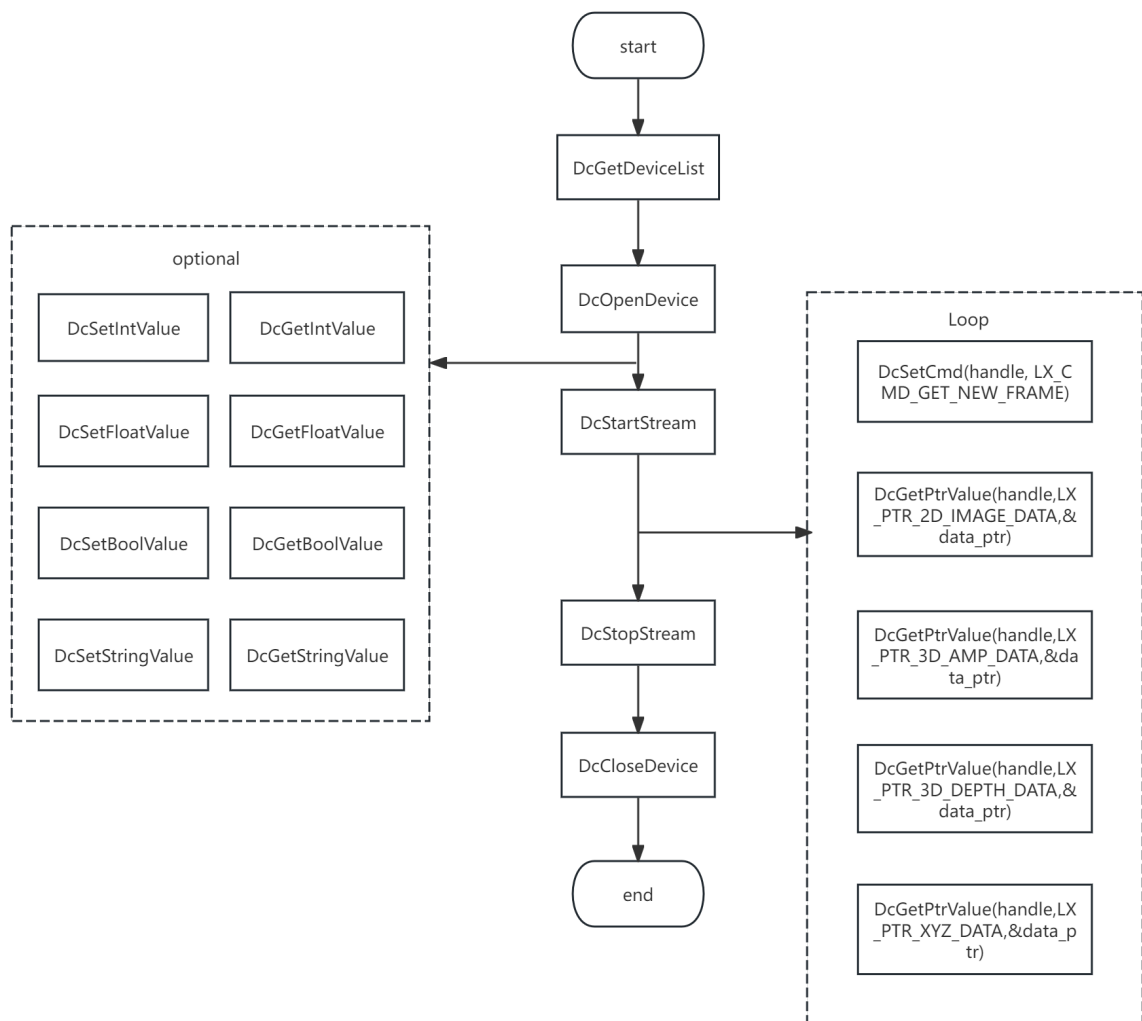
SDK: contains header files and library files for the Camera SDK

3.2 Development process

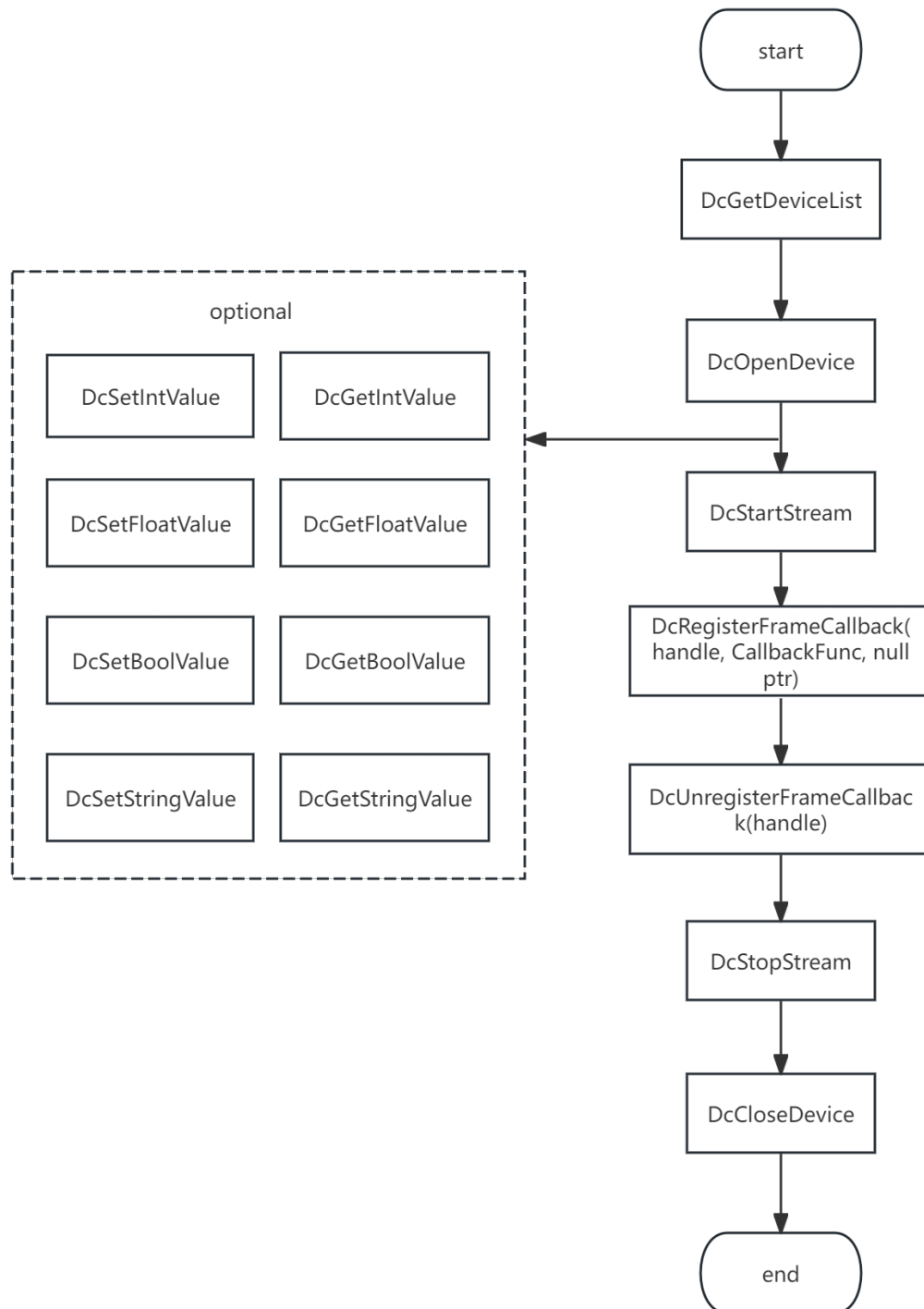
3.2.1 Project setup

C/C++ is recommended as the programming language for development based on the SDK, and CMakeLists.txt scripts are provided for building projects using the CMAKE tool. The Sample directory provides examples of development methods for different programming languages. For details, please refer to the materials in Sample directory and "Instructions for Linux Sample Programs".

3.2.2 Call APIs synchronously



3.2.3 Call APIs asynchronously



4 Precautions for calling the Camera SDK

4.1 Principles for calling the Camera SDK

Use `DcSetIntValue` or `DcGetIntValue` to set or obtain INT type parameters for enumeration type prefixed with `LX_INT`.

Use `DcSetFloatValue` or `DcGetFloatValue` to set or obtain FLOAT type parameters for enumeration type prefixed with `LX_FLOAT`.

Use `DcSetBoolValue` or `DcGetBoolValue` to set or obtain BOOL type parameters for enumeration type prefixed with `LX_BOOL`.

Use `DcSetStringValue` or `DcGetStringValue` to set or get String type parameters for enumeration type prefixed with `LX_STRING`.

Use `DcSetCmd` to execute instructions for enumeration type prefixed with `LX_CMD`.

Use `DcGetPtrValue` to obtain the pointer type result for enumeration type prefixed with `LX_PTR`.

4.2 Prerequisites for calling the Camera SDK interfaces

In start-stream state or the work mode (`LX_INT_WORK_MODE`) is `WORK_FOREVER`, the following settings are not allowed:

`DcSetRoi`

`LX_INT_3D_BINNING_MODE;`

`LX_INT_2D_BINNING_MODE;`

`LX_BOOL_ENABLE_2D_TO_DEPTH;`

`LX_BOOL_ENABLE_MULTI_EXPOSURE_HDR;`

`LX_BOOL_ENABLE_MULTI_MACHINE;`

`LX_INT_TRIGGER_MODE;`

`LX_INT_CALCULATE_UP;`

`LX_BOOL_ENABLE_HDR`

LX_BOOL_ENABLE_TOF_DEPTH_CALIBE

LX_BOOL_ENABLE_SYNC_FRAME

LX_INT_SAVE_PARAMS_GROUP

When the work mode (LX_INT_WORK_MODE) is WORK_FOREVER, the following setting is not allowed:

LX_INT_TOF_FREQMODE

When 3D Auto Exposure (LX_BOOL_ENABLE_3D_AUTO_EXPOSURE) is enabled, the following function enumerations are not allowed:

LX_BOOL_ENABLE_MULTI_EXPOSURE_HDR;

LX_INT_FIRST_EXPOSURE;

LX_INT_SECOND_EXPOSURE;

LX_INT_THIRD_EXPOSURE;

LX_INT_GAIN;

When 3D Auto Exposure (LX_BOOL_ENABLE_3D_AUTO_EXPOSURE) is disabled, the following function enumeration is not allowed:

LX_INT_3D_AUTO_EXPOSURE_LEVEL

And setting the maximum (LX_INT_3D_AUTO_EXPOSURE_MAX) and minimum (LX_INT_3D_AUTO_EXPOSURE_MIN) values of 3D Auto Exposure is not allowed

When 2D Auto Exposure (LX_BOOL_ENABLE_2D_AUTO_EXPOSURE) is enabled, the following function enumeration is not allowed:

LX_INT_2D_MANUAL_EXPOSURE

When 2D Auto Exposure (LX_BOOL_ENABLE_2D_AUTO_EXPOSURE) is disabled, the following function enumeration is not allowed:

LX_INT_2D_AUTO_EXPOSURE_LEVEL;

When the filter mode is not FILTER_NORMAL, the following settings are not allowed:

LX_INT_FILTER_SMOOTH_LEVEL

LX_INT_FILTER_NOISE_LEVEL

LX_INT_FILTER_TIME_LEVEL

Enable RGBD alignment before enabling 2D undistort

LX_BOOL_ENABLE_2D_TO_DEPTH

For devices other than S2 type, set the tof calculation down before enabling the algorithm.

After the algorithm is enabled (LX_INT_ALGORITHM_MODE) it is allowed to set the algorithm parameters (LX_STRING_ALGORITHM_PARAMS).

To obtain point cloud (LX_PTR_XYZ_DATA), depth image must be enabled first (LX_BOOL_ENABLE_3D_DEPTH_STREAM).

5 Data type specification

5.1 LX_STATE

Feature: The status code of the value returned by the SDK interface function

Keywords	Specifications
LX_SUCCESS	The function is executed successfully
LX_ERROR	The function failed to be executed
LX_E_NOT_SUPPORT	This feature is not supported by the connected camera

LX_E_NETWORK_ERROR	Network communication error
LX_E_INPUT_ILLEGAL	Illegal function argument passed in
LX_E_RECONNECTING	The device is being reconnected
LX_E_DEVICE_ERROR	Equipment failure
LX_E_DEVICE_NEED_UPDATE	The device firmware needs to be upgraded as the current version is too low
LX_E_API_NEED_UPDATE	The SDK needs to be upgraded as the API version is too low
LX_E_CTRL_PERMISS_ERROR	Failed to get exclusive control permission (only one SDK can connect to a device at a time)
LX_E_GET_DEVICEINFO_ERROR	Failed to obtain device information
LX_E_IMAGE_SIZE_ERROR	Image size error, it needs to reboot the camera
LX_E_IMAGE_PARTITION_ERROR	Failed to parse the image
LX_E_DEVICE_NOT_CONNECTED	The camera is not connected
LX_E_DEVICE_INIT_FAILED	Failed to initialize the camera
LX_E_DEVICE_NOT_FOUND	Failed to find the camera
LX_E_FILE_INVALID	File error (file name or type or format is incorrect)
LX_E_CRC_CHECK_FAILED	Failed to complete the crc or md5 verification
LX_E_TIME_OUT	Timeout
LX_E_FRAME_LOSS	One or more frames are missing
LX_E_ENABLE_ANYSTREAM_FAILED	Failed to enable any stream
LX_E_NOT_RECEIVE_STREAM	Stream not received
LX_E_PARSE_STREAM_FAILED	Start steaming is successful but failed to parse the stream
LX_E_PROCESS_IMAGE_FAILED	Failed to process the image data
LX_E_SETTING_NOT_ALLOWED	Not allowed to set in WORK_FOREVER mode
LX_E_LOAD_DATAPROCESSLIB_ERROR	Error loading image processing algorithm library

LX_E_FUNCTION_CALL_LOGIC_ERROR	Logical error when calling the function
LX_E_IPAPPDR_UNREACHABLE_ERROR	IP is unreachable or network configuration error
LX_E_FRAME_ID_NOT_MATCH	Frame out of sync error within timeout range
LX_E_FRAME_MULTI_MACHINE	A multi-camera jamming signal was detected in the frame

5.2 LX_DATA_TYPE

Feature: Data format. 3D amplitude images and depth images have different data formats, and you need to use the corresponding feature to obtain data pointers.

Keywords	Specifications
LX_DATA_UNSIGNED_CHAR	Unsigned characters
LX_DATA_UNSIGNED_SHORT	Unsigned short integer
LX_DATA_SIGNED_SHORT	Signed short integer
LX_DATA_FLOAT	Floating point
LX_DATA_OBSTACLE	The structure type of the obstacle avoidance algorithm
LX_DATA_PALLET	The structure type of the pallet algorithm
LX_DATA_LOCATION	The type of structure of the visual positioning algorithm
LX_DATA_OBSTACLE2	The structure type of the obstacle avoidance algorithm V2

5.3 LX_BINNING_MODE

Feature: binning mode

Keywords	Specifications
----------	----------------

LX_BINNING_1X1	1x1Binning
LX_BINNING_2X2	2x2Binning
LX_BINNING_4X4	4x4Binning

5.4 LX_STRUCT_LIGHT_CODE_MODE

Feature:

Keywords	Specifications
LX_CODE_NORMAL	Normal
LX_CODE_STATBLE	Stable
LX_CODE_ENHANCE	Enhanced with high precision

5.5 LX_ALGORITHM_MODE

Feature:

Keywords	Specifications
MODE_ALL_OFF	Turn off the camera's built-in algorithms
MODE_AVOID_OBSTACLE	Built-in obstacle avoidance algorithm
MODE_PALLET_LOCATE	Built-in pallet algorithm
MODE_VISION_LOCATION	Built-in visual positioning algorithm
MODE_AVOID_OBSTACLE2	Built-in obstacle avoidance algorithm V2

5.6 LX_CAMERA_WORK_MODE

Feature:

Keywords	Specifications
KEEP_HEARTBEAT	Heartbeat mode, the camera enters standby mode after the SDK heartbeat is interrupted.
WORK_FOREVER	Forever mode, the camera is always in working condition

5.7 LX_TRIGGER_MODE

Feature:

Keywords	Specifications
LX_TRIGGER_MODE_OFF	Trigger mode is turned off, streaming mode is activated as default.
LX_TRIGGER_SOFTWARE	Software trigger mode
LX_TRIGGER_HARDWARE	Hardware trigger mode

5.8 LX_CAMERA_FEATURE

Feature: various readable and writable parameters

5.8.1 INT type parameters

Common parameters

Keywords	Specifications
LX_INT_FIRST_EXPOSURE	Exposure time, in us, the exposure time of the first integral for the multi-integration case.
LX_INT_SECOND_EXPOSURE	Exposure time of the second integral for the multi-integration case.
LX_INT_THIRD_EXPOSURE	Exposure time of the third integral for the multi-integration case, only supported by some of the models.
LX_INT_FOURTH_EXPOSURE	Exposure time of the forth integral for the multi-integration case, only supported by some of the models.
LX_INT_GAIN	Gain, which is equivalent to the exposure effect. Noise will be introduced, and the gain can be adjusted appropriately to prevent the exposure

	parameters from being too large.
LX_INT_MIN_DEPTH	Minimum depth value, values out of range are invalid.
LX_INT_MAX_DEPTH	Maximum depth value, values out of range are invalid.
LX_INT_MIN_AMPLITUDE	Minimum amplitude value of the valid signal, values out of range are invalid.
LX_INT_MAX_AMPLITUDE	Maximum amplitude value of the valid signal, values out of range are invalid.
LX_INT_CODE_MODE	Structured light camera encoding mode, refer to CODE_MODE (supported by some models).
LX_INT_WORK_MODE	Work mode, refer to LX_CAMERA_WORK_MODE
LX_INT_LINK_SPEED	The negotiated network speed is 100-100 Gigabit, 1000-Gigabit, readable only.
LX_INT_TOF_GLOBAL_OFFSET	TOF depth data offset
LX_INT_ALGORITHM_MODE	Built-in algorithm is enabled, only some models can support it, refer to LX_ALGORITHM_MODE for corresponding values.
LX_INT_TRIGGER_MODE	Trigger mode, refer to LX_TRIGGER_MODE for corresponding values.
LX_INT_MODBUS_ADDR	ModBus address, only some models can support ModBus output via serial port.
LX_INT_HEART_TIME	Heartbeat time, the camera will automatically standby if the heartbeat is not received after timeout.
LX_INT_GVSP_PACKET_SIZE	The size of a GVSP packet in bytes
LX_INT_CALCULATE_UP	The TOF or RGB algorithm is allowed to move up and down, saving the computing power of the host computer or camera, which may affect the

	frame rate and latency.
LX_INT_CAN_BAUD_RATE	The baud rate value of CAN, in bps
LX_INT_SAVE_PARAMS_GROUP	Saves the current configuration of the camera as a specified parameter group
LX_INT_LOAD_PARAMS_GROUP	One-click loading of parameter groups for specified indexes

3D Image parameters

Keywords	Specifications
LX_INT_3D_IMAGE_WIDTH	The current width of the 3D image resolution
LX_INT_3D_IMAGE_HEIGHT	The current height of the 3D image resolution
LX_INT_3D_IMAGE_OFFSET_X	ROI horizontal offset pixels, use DcSetRol to set the parameters.
LX_INT_3D_IMAGE_OFFSET_Y	ROI vertical offset pixels, use DcSetRol to set the parameters.
LX_INT_3D_BINNING_MODE	Pixel binning, refer to LX_BINNING_MODE
LX_INT_3D_DEPTH_DATA_TYPE	Depth data format, readable only, refer to LX_DATA_TYPE for corresponding values.
LX_INT_3D_AMPLITUDE_CHANNEL	Channels of 3D amplitude images, shared with depth images.
LX_INT_3D_AMPLITUDE_DATA_TYPE	Amplitude data format, readable only, refer to LX_DATA_TYPE for corresponding values.
LX_INT_3D_AUTO_EXPOSURE_LEVEL	When 3D Auto Exposure is enabled, the exposure value and gain cannot be set.
LX_INT_3D_AUTO_EXPOSURE_MAX	The upper limit of 3D auto exposure
LX_INT_3D_AUTO_EXPOSURE_MIN	The lower limit of 3D auto exposure
LX_INT_3D_UNDISTORT_SCALE	3D image undistort coefficient

2D Image parameters

Keywords	Specifications
LX_INT_2D_IMAGE_CHANNEL	Number of channels for 2D images, 1 for monochrome images and 3 for color images
LX_INT_2D_IMAGE_WIDTH	The current width of the 2D image resolution
LX_INT_2D_IMAGE_HEIGHT	The current height of the 2D image resolution
LX_INT_2D_IMAGE_OFFSET_X	2D ROI horizontal offset pixels, use DcSetRol to set the parameters
LX_INT_2D_IMAGE_OFFSET_Y	2D ROI vertical offset pixels, use DcSetRol to set the parameters
LX_INT_2D_BINNING_MODE	Pixel binning, refer to LX_BINNING_MODE
LX_INT_2D_MANUAL_EXPOSURE	Exposure value in manual exposure for 2D images
LX_INT_2D_MANUAL_GAIN	Gain value in manual exposure for 2D images
LX_INT_2D_AUTO_EXPOSURE_LEVEL	Exposure level in auto-exposure for 2D images
LX_INT_2D_IMAGE_DATA_TYPE	2D image data format, readable only, refer to LX_DATA_TYPE for corresponding values.
LX_INT_2D_UNDISTORT_SCALE	2D image undistort coefficient

Trigger parameters

Keywords	Specifications
LX_INT_TRIGGER_MODE	Trigger mode, refer to LX_TRIGGER_MODE for corresponding values.
LX_INT_HARDWARE_TRIGGER_FILTER_TIME	Filter time for hardware trigger, in us
LX_INT_TRIGGER_MIN_PERIOD_TIME	Minimum time interval for triggering, in us
LX_INT_TRIGGER_DELAY_TIME	Trigger delay time, in us, when the value is <=1000, it takes effect immediately (reserved

	function, if it is greater than 1000, it takes effect after the delay).
LX_INT_TRIGGER_FRAME_COUNT	The number of frames triggered at a time
LX_INT_IO_WORK_MODE	GPIO signal output control mode, refer to LX_IO_WORK_MODE
LX_INT_IO_OUTPUT_STATE	User control mode for GPIO signal output, refer to LX_IO_OUTPUT_STATE

Filter parameters

Keywords	Specifications
LX_INT_FILTER_MODE	Filter mode, refer to LX_FILTER_MODE
LX_INT_FILTER_SMOOTH_LEVEL	When LX_INT_FILTER_MODE is FILTER_NORMAL, you can set the filter smooth level within [0, 3], the higher the value, the stronger the filtering.
LX_INT_FILTER_NOISE_LEVEL	When LX_INT_FILTER_MODE is FILTER_NORMAL, you can set the filter noise level within [0, 3], the higher the value, the stronger the filtering.
LX_INT_FILTER_TIME_LEVEL	When LX_INT_FILTER_MODE is FILTER_NORMAL, you can set the filter time level within [0, 3], the higher the value, the stronger the filtering.

5.8.2 Float type parameters

Keywords	Specifications
LX_FLOAT_FILTER_LEVEL	Filter level, 0-1, the higher the value, the stronger the filtering.
LX_FLOAT_EST_OUT_EXPOSURE	Estimate whether it is over exposure or not, 0-1, if it is 1 then it is over exposure and the data is invalid.
LX_FLOAT_LIGHT_INTENSITY	Light intensity, only supported by some models.

LX_FLOAT_3D_DEPTH_FPS	Current frame rate of depth images
LX_FLOAT_3D_AMPLITUDE_FPS	Current frame rate of amplitude images
LX_FLOAT_2D_IAMGE_FPS	Current frame rate of RGB images
LX_FLOAT_DEVICE_TEMPERATURE	Obtain the current temperature of the camera

5.8.3 Bool type parameters

Keywords	Specifications
LX_BOOL_CONNECT_STATE	Current connection state
LX_BOOL_ENABLE_3D_DEPTH_STREAM	Enable/disable depth image steaming (some of the models support it)
LX_BOOL_ENABLE_3D_AMP_STREAM	Enable/disable amplitude image steaming (some of the models support it)
LX_BOOL_ENABLE_3D_AUTO_EXPOSURE	Enable 3D auto exposure
LX_BOOL_ENABLE_3D_UNDISTORT	Enable 3D undistort
LX_BOOL_ENABLE_BACKGROUND_AMP	Enable background light amplitude
LX_BOOL_ENABLE_ANTI_FLICKER	Enable anti-flicker, LED ambient lighting may cause obvious ripples in the data, and some of the models support it.
LX_BOOL_ENABLE_2D_STREAM	Enable/disable RGB
LX_BOOL_ENABLE_2D_AUTO_EXPOSURE	Enable 2D auto exposure
LX_BOOL_ENABLE_2D_UNDISTORT	Enable 2D undistort
LX_BOOL_ENABLE_2D_TO_DEPTH	Enable RGBD coordinate alignment
LX_BOOL_ENABLE_MULTI_MACHINE	Enable multiple cameras, and some of the models support it.
LX_BOOL_ENABLE_MULTI_EXPOSURE_HDR	Enable HDR (high dynamic range)
LX_BOOL_ENABLE_SYNC_FRAME	Whether to enable forced frame synchronization or not, by default data real-time is prioritized, if RGBD synchronization is required, this mode needs to be enabled.

5.8.4 String type parameters

Keywords	Specifications
LX_STRING_DEVICE_VERSION	Device version
LX_STRING_FIRMWARE_NAME	Firmware name, used to upgrade the device, writeable only.
LX_STRING_FILTER_PARAMS	Filter parameters, string in json format.
LX_STRING_ALGORITHM_PARAMS	Built-in algorithm parameters, different starting modes have different JSON format strings, LX_ALGORITHM_MODE needs to be configured in advance.
LX_STRING_ALGORITHM_VERSION	The version number of the built-in algorithm. For different starting modes it returns corresponding versions, readable only, LX_ALGORITHM_MODE needs to be configured in advance.
LX_STRING_DEVICE_OS_VERSION	The version number of the device system image
LX_STRING_IMPORT_PARAMS_FROM_FILE	Load parameters from a local file to the camera
LX_STRING_EXPORT_PARAMS_TO_FILE	Export the current parameters of the camera to a local file

5.8.5 Command type parameters

Keywords	Specifications
LX_CMD_GET_NEW_FRAME	Get a new frame data, the default timeout period is 1s.
LX_CMD_RETURN_VERSION	Return to the last version
LX_CMD_RESTART_DEVICE	Restart the camera

LX_CMD_WHITE_BALANCE	Auto white balance
LX_CMD_RESET_PARAM	Restore to the default parameters

5.8.6 Ptr type parameters

Keywords	Specifications
LX_PTR_2D_IMAGE_DATA	Obtain the pointer to the 2D image, and the data length is determined by the 2D image size, number of channels, and data format (LX_INT_2D_IMAGE_DATA_TYPE).
LX_PTR_3D_AMP_DATA	Obtain the pointer to the 3D amplitude image, and the data length is determined by the 3D image size, number of channels, and data format (LX_INT_3D_AMPLITUDE_DATA_TYPE).
LX_PTR_3D_DEPTH_DATA	Obtain the pointer to the 3D depth image, the data length is determined by the 3D image size, number of channels, and data format (LX_INT_3D_DEPTH_DATA_TYPE).
LX_PTR_XYZ_DATA	Obtain the pointer to the point cloud, float* type for three channels (x, y, z are a set of data, cyclic in turn), and the data length is LX_INT_3D_IMAGE_WIDTH*LX_INT_3D_IMAGE_HEIGHT*sizeof(float)*3.
LX_PTR_ALGORITHM_OUTPUT	Get the output of the built-in algorithm, which is related to the algorithm mode. MODE_AVOID_OBSTACLE is the pointer to LxAvoidanceOutput, MODE_PALLET_LOCATE is the pointer to LxPalletPose, and MODE_VISION_LOCATION is the pointer to the reference LxLocation.
LX_PTR_2D_INTRIC_PARAM	Obtain the intrinsic parameters of the 2D image, float* type pointer, the length is fixed which is 9*sizeof(float)(fx,fy,cx,cy,k1,k2,p1,p2,k3).
LX_PTR_3D_INTRIC_PARAM	Obtain the intrinsic parameters of the 3D image, float*

	type pointer, the length is fixed which is $9 * \text{sizeof}(\text{float})(fx, fy, cx, cy, k1, k2, p1, p2, k3)$.
LX_PTR_3D_EXTRIC_PARAM	Obtain the extrinsic parameters of the 3D image, float* type pointer, the length is fixed which is $12 * \text{sizeof}(\text{float})$ (the first 9 represent the rotation matrix, and the last 3 represent the translation vector).
LX_PTR_FRAME_DATA	Obtain a complete frame data, and refer to FrameInfo for output result.

5.9 LxDeviceInfo

Feature: Device attribute structure

Keywords	Type	Specifications
handle	DCHandle	Unique device identifier
dev_type	LX_DEVICE_TYPE	Device type
id	Char array	Device id
ip	Char array	Device ip:port
sn	Char array	The serial number of the device
mac	Char array	The MAC address of the device
firmware_ver	Char array	The software version number of the device
algor_ver	Char array	The algorithm version number of the device
name	Char array	Device name , for example: camera_M3_192.168.11.13_9803
reserve	Char array	Reserved field, subnet mask
reserve2	Char array	Reserved field 2, Gateway
reserve3	Char array	Reserved field 3
reserve4	Char array	Reserved field 4

5.10 LxIntValueInfo

Feature: Device attribute structure

Keywords	Type	Specifications
set_available	Bool	Whether the current value can be set or not, true – can be set, false - cannot be set.
cur_value	Int	current value
max_value	Int	Maximum value
min_value	Int	Minimum value
reserve	Int array	Reserved field

5.11 LxFloatValueInfo

Feature: Device attribute structure

Keywords	Type	Specifications
set_available	Bool	Whether the current value can be set or not, true – can be set, false - cannot be set.
cur_value	float	current value
max_value	float	Maximum value
min_value	float	Minimum value
reserve	Float array	Reserved field

5.12 LX_DEVICE_TYPE

Feature: Device type

Keywords	Specifications
LX_DEVICE_M2	M2 camera
LX_DEVICE_M3	M3 camera
LX_DEVICE_M4Pro	M4Pro camera
LX_DEVICE_M4_MEGA	M4MEGA camera
LX_DEVICE_M4	M4 camera
LX_DEVICE_S1	S1 camera
LX_DEVICE_S2	S2 camera
LX_DEVICE_I1	I1 camera
LX_DEVICE_I2	I2 camera
LX_DEVICE_T1	T1 camera
LX_DEVICE_T2	T2 camera
LX_DEVICE_H3	H3 camera
LX_DEVICE_V1Pro	V1Pro camera
LX_DEVICE_NULL	Reserved

5.13 LX_OPEN_MODE

Feature: Device type

Keywords	Specifications
OPEN_BY_INDEX	Open the device using the index subscript in the search list, and the corresponding parameter is the index number. When the list of searched devices changes, the devices selected to open will be different.
OPEN_BY_IP	Open the device using the corresponding IP address in the search

	list, and the corresponding parameter is the device IP or IP:port.
OPEN_BY_SN	Open the device using the corresponding sn in the search list, and the corresponding parameter is the device sn.
OPEN_BY_ID	Open the device using the corresponding id in the search list, and the corresponding parameter is the device id.

5.14 FrameDataInfo

Feature: Image display structure

Keywords	Type	Specifications
frame_data_type	LX_DATA_TYPE	Frame data type
frame_width	Int	The image width of the frame data
frame_height	Int	The image height of the frame data
frame_channel	Int	The number of channels of the frame data
frame_data	Void * pointer type	The content of the frame data
sensor_timestamp	Unsigned long long type	The timestamp of the sensor plot
recv_timestamp	Unsigned long long type	The timestamp when the frame data was received

5.15 FrameInfo

Feature: Frame data structure

Keywords	Type	Specifications
frame_state	LX_STATE	Frame data state
handle	DcHandle	Unique device identifier
depth_data	FrameDataInfo type	Frame data structure of depth image

amp_data	FrameDataInfo type	Frame data structure of amplitude image
rgb_data	FrameDataInfo type	Frame data structure of RGB image
app_data	FrameDataInfo type	Structure of algorithm output
reserve_data	Void *pointer type	Reserved field

6 API specification

6.1 Find, connect, and turn off the camera

6.1.1 DcGetDeviceList

Function archetype:

```
#ifndef __cplusplus
LX_API DcGetDeviceList(LxDeviceInfo** devlist, int* devnum, LX_DEVICE_SERIALS lx_serials = LX_SERIAL_ALL);
#else
//纯C风格接口，原接口逐步弃用
LX_API DcGetDeviceList(LxDeviceInfo** devlist, int* devnum);
#endif
```

Function feature:

Get a list of supported cameras.

Function parameters:

[out]devlist List of cameras found
 [out]devnum Number of cameras found

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: Memory is maintained internally, and no memory needs to be allocated externally, but thread safety needs to be ensured externally because the original content will be cleared before each search.

6.1.2 DcOpenDevice

Function archetype:

```
LX_API DcOpenDevice(LX_OPEN_MODE open_mode, const char* param, DCHandle* handle, LxDeviceInfo* info);
```

Function feature:

Connect to devices

Function parameters:

[in]open_mode The way to open the device, for more information, refer to OpenMode.
 [in]param Different way to open the device requires different parameters to be written.

[out]handle The handle of the device returned after the connection is successful, and all subsequent interface access depends on the handle.

[out]info The camera details returned after a successful connection

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid.

6.1.3 DcCloseDevice

Function archetype:

```
LX_API DcCloseDevice(DCHandle handle);
```

Function feature:

Close the device

Function parameters:

[in]handle device handle

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid.

6.2 Start and stop data stream

6.2.1 DcStartStream

Function archetype:

```
LX_API DcStartStream(DCHandle handle);
```

Function feature:

Start the data stream

Function parameters:

[in]handle device handle

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid.

6.2.2 DcStopStream

Function archetype:

```
LX_API DcStopStream(DCHandle handle);
```

Function feature:

Stop the data stream

Function parameters:

[in]handle device handle

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid.

6.3 Get and set the camera parameters

6.3.1 DcSetIntValue

Function archetype:

```
LX_API DcSetIntValue(DcHandle handle, int cmd, int value);
```

Function feature:

Set int type parameters

Function parameters:

[in]handle	device handle
[in]cmd	refer to LX_CAMERA_FEATURE
[in]value	Set the parameter values

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid.

6.3.2 DcGetIntValue

Function archetype:

```
LX_API DcGetIntValue(DcHandle handle, int cmd, LxIntValueInfo* value);
```

Function feature:

Get int type parameters

Function parameters:

[in]handle	device handle
[in]cmd	refer to LX_CAMERA_FEATURE
[out]value	return structure parameters with

Maximum / minimum and current values

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid.

6.3.3 DcSetFloatValue

Function archetype:

```
LX_API DcSetFloatValue(DCHandle handle, int cmd, float value);
```

Function feature:

Set float type parameters

Function parameters:

[in]handle	device handle
[in]cmd	refer to LX_CAMERA_FEATURE
[in]value	set the parameter values

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid.

6.3.4 DcGetFloatValue

Function archetype:

```
LX_API DcGetFloatValue(DCHandle handle, int cmd, LxFloatValueInfo* value);
```

Function feature:

Get float type parameters

Function parameters:

[in]handle	device handle
[in]cmd	refer to LX_CAMERA_FEATURE
[out]value	return structure parameters with

Maximum / minimum and current values

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid.

6.3.5 DcSetBoolValue

Function archetype:

```
LX_API DcSetBoolValue(DCHandle handle, int cmd, bool value);
```

Function feature:

Set bool type parameters

Function parameters:

[in]handle	device handle
[in]cmd	refer to LX_CAMERA_FEATURE
[in]value	set the parameter values

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid.

6.3.6 DcGetBoolValue

Function archetype:

```
LX_API DcGetBoolValue(DCHandle handle, int cmd, bool* value);
```

Function feature:

Get bool type parameters

Function parameters:

[in]handle	device handle
[in]cmd	refer to LX_CAMERA_FEATURE
[out]value	return values

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid.

6.3.7 DcSetStringValue

Function archetype:

```
LX_API DcSetStringValue(DCHandle handle, int cmd, const char* value);
```

Function feature:

Set string type parameters

Function parameters:

[in]handle	device handle
[in]cmd	refer to LX_CAMERA_FEATURE

[in]value set values

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid.

6.3.8 DcGetStringValue

Function archetype:

```
LX_API DcGetStringValue(DcHandle handle, int cmd, char** value);
```

Function feature:

Get string type parameters

Function parameters:

[in]handle device handle

[in]cmd refer to LX_CAMERA_FEATURE

[out]value return values, without memory allocation
externally

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid.

6.3.9 DcGetPtrValue

Function archetype:

```
LX_API DcGetPtrValue(DcHandle handle, int cmd, void** value);
```

Function feature:

Get pointer type parameters

Function parameters:

[in]handle device handle

[in]cmd refer to LX_CAMERA_FEATURE

[out]value return values, without memory allocation
externally

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid.

6.3.10 DcSetCmd

Function archetype:

```
LX_API DcSetCmd(DCHandle handle, int cmd);
```

Function feature:

Execute CMD type instructions

Function parameters:

[in]handle device handle

[in]cmd refer to LX_CAMERA_FEATURE

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid.

6.4 Save point cloud

6.4.1 DcSaveXYZ

Function archetype:

```
LX_API DcSaveXYZ(DCHandle handle, const char* filename);
```

Function feature:

Save point cloud, and can be called directly

Function parameters:

[in]handle device handle

[in]filename File name, supports txt, ply, and pcd formats. The TXT format saves all the data in image order, and PLY and PCD only save non-zero data

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid.

6.5 Special control

6.5.1 DcSpecialControl

Function archetype:

```
LX_API DcSpecialControl(DcHandle handle, const char* command, void* value);
```

Function feature:

Special operations outside of LX_CAMERA_FEATURE definition

Function parameters:

[in]handle device handle

[in]param operation

[inout]value When set, it is the corresponding input parameter, and when it is obtained, it is the corresponding output parameter, and there is no need to allocate memory externally

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid.

6.6 Set the ROI and parameter paths

6.6.1 DcSetRoi

Function archetype:

```
LX_API DcSetROI(DcHandle handle, int offsetx, int offsety, int width, int height, int img_type);
```

Function feature:

Set the ROI field, if the input value is not an integer multiple of 8, the integer multiple of the nearest 8 will be automatically processed internally. After setting, you need to update the image size parameters

Function parameters:

[in]handle device handle

[in]offsetx	horizontal pixel offset for start point
[in]offsety	vertical pixel offset for start point
[in]width	width of ROI
[in]height	height of ROI
[in]type	0-3Dimage 1-2Dimage

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid.

6.7 Set camera IP

6.7.1 DcSetCameraIp

Function archetype:

```

#ifdef __cplusplus
LX_API DcSetCameraIp DCHandle handle, const char* ip, const char* netmask = 0, const char* gateway = 0;
#else
//纯C风格接口，原接口逐步弃用
LX_API DcSetCameraIp DCHandle handle, const char* ip, const char* netmask, const char* gateway);
#endif
  
```

Function feature:

Set the camera IP and subnet mask, gateway

Function parameters:

[in]handle	device handle
[in]ip	device IP
[in]netmask	Subnet mask (if empty, the internal default is "255.255.0.0")
[in]gateway	Gateway IP address (if empty, the last CIDR block of the IP address is set to "1" by default as the gateway)

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: If the device is not connected, you need to search for it. If the device supports modifying the gateway and subnet masks, set the gateway to the same CIDR block

and set the subnet mask to 255.255.0.0. After the modification is complete, the device list will change, and you need to call the DcGetDeviceList operation again to obtain the new device list.

6.8 Log and version information

6.8.1 DcGetApiVersion

Function archetype:

```
LX_API_STR DcGetApiVersion();
```

Function feature:

Get the API version

Function parameters:

Return value: char type pointer data

Note:

6.8.2 DcSetInfoOutput

Function archetype:

```
#ifndef __cplusplus
LX_API DcSetInfoOutput(int print_level = 2, bool enable_screen_print = false, const char* log_path = "", int language = 0);
#else
// 纯C风格接口、原接口逐步弃用
LX_API DcSetLog(int print_level, bool enable_screen_print, const char* log_path);
#endif
```

Function feature:

Set the level of print information

Function parameters:

[in]print_level 0: info for all debugs, 1: warn important and warning debugs, 2: error for all error messages

[in]enable_screen_print if to print on screen

[in]log_path The log file path can be empty (not include log file name).

The default is in the current user path for Windows, and the /var/log path is for Linux

[in]language language (not used now)

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid

6.9 DcLog

Function archetype:

```
LX_API DcLog(const char* str);
```

Function feature:

Allows the user to output debug information to a log file

Function parameters:

[in]str the string to be output, ending with '\0'

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid

6.9.1 DcRegisterFrameCallback

Function archetype:

```
#ifdef __cplusplus
LX_API DcRegisterFrameCallback DcHandle handle, LX_FRAME_CALLBACK func, void* usr_data = 0);
#else
/*纯C风格接口，原接口逐步弃用*/
LX_API DcRegisterFrameCallback DcHandle handle, LX_FRAME_CALLBACK func, void* usr_data);
#endif
```

Function feature:

Callback function for dataframe registration and will be automatically called when new data is received

Function parameters:

[in]handle device handle

[in]func callback function for dataframe

[in]usr_data user defined parameter, can be empty

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid

6.9.2 DcUnregisterFrameCallback

Function archetype:

```
LX_API DcUnregisterFrameCallback(DCHandle handle);
```

Function feature:

Cancel the frame callback

Function parameters:

[in]handle device handle

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid

6.10 Device status callback

6.10.1 DcRegisterCameraStatusCallback

Function archetype:

```
LX_API DcRegisterCameraStatusCallback(DCHandle handle, LX_CAMERA_STATUS_CALLBACK func, void* usr_data)
```

Function feature:

Callback function for camera status registration, which is automatically called when the camera status changes

Function parameters:

[in]handle device handle

[in]func callback function for dataframe

[in]usr_data user defined parameter, can be empty

Note: ensure that the parameters passed in are valid

6.10.2 DcUnregisterCameraStatusCallback

Function archetype:

```
LX_API·DcUnregisterCameraStatusCallback(DcHandle·handle)
```

Function feature:

Cancel callback for camera status registration

Function parameters:

[in]handle device handle

Return value:

Int data, if this value equals to LX_SUCCESS, it means API is called successfully.

Note: ensure that the parameters passed in are valid

7 Versions and updates

7.1 The latest versions and updates

Versions	Change notes
2.4.16	1.Support S2 Max camera 2.Add parameter import and export and parameter group switching 3.Added relevant function settings
2.4.9	1.Optimize and add some functions and configuration parameters 2.Change the timestamp to us
2.0.2	Fix some issues and adjust some interfaces and parameters