

Java Interface Development User Manual

All rights reserved. No parts of this manual may be used or reproduced, in any forms or by any means, without prior written permission of China Daheng Group, Inc. Beijing Image Vision Technology Branch.

The right is also reserved to modify or change any parts of this manual in the future without prior notification.

All other trademarks are the properties of their respective owners.

© 2019 China Daheng Group, Inc. Beijing Image Vision Technology Branch

Web: <http://www.daheng-imaging.com>

Sales Email: isales@daheng-imaging.com

Sales Tel: +86 10 8282 8878

Support Email: isupport@daheng-imaging.com

Contents

1. Camera Workflow	1
1.1. Overall workflow	1
1.1.1. General enumeration method.....	1
1.1.2. Enumeration without Root permission	2
1.1.3. Selection of enumeration methods.....	2
1.2. Image acquisition flow	3
1.2.1. getRawImage method	3
1.2.2. getBitmap method	4
1.2.3. getImageBySurface method.....	5
1.3. Function control flow	6
1.4. Overall code sample.....	6
1.4.1. General enumeration method.....	6
1.4.2. Enumeration without Root permission	7
2. Build Environment.....	9
2.1. Android	9
2.1.1. Create project	9
2.1.2. Importing library	12
2.1.3. javadoc configuring.....	14
3. Programming Guide	16
3.1. Importing library.....	16
3.2. Enumeration device.....	16
3.2.1. General method	17
3.2.2. No Root permission method	17
3.3. Open the device	18
3.4. Acquisition control	19
3.5. Image acquisition and processing	19
3.5.1. getRawImage method	20
3.5.2. getBitmap method	22
3.5.3. getImageBySurface method.....	24
3.5.4. Image quality improvement	27
3.6. Camera control.....	28
3.6.1. Feature parameter access type.....	28
3.6.2. Feature control.....	28
3.7. Import and export camera configuration	31
3.8. Exceptions.....	31
3.8.1. Sample code.....	31
3.8.2. Exception type	32
4. FAQ	34
5. Appendix	35

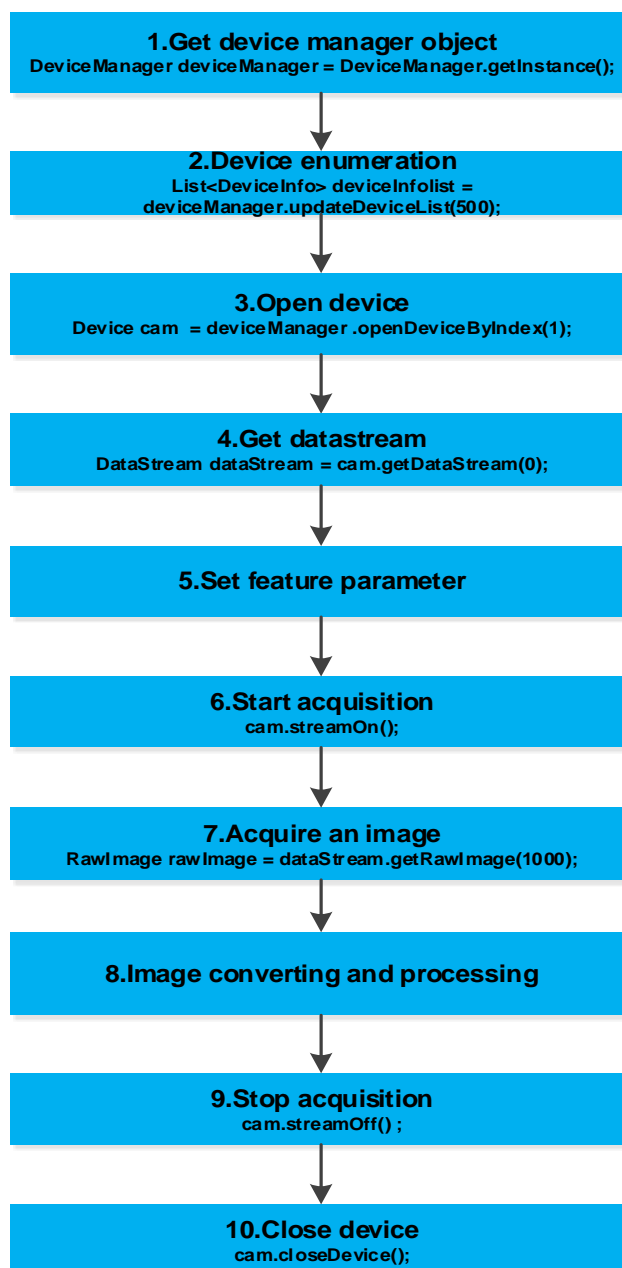
5.1. Feature parameter.....	35
5.1.1. Device feature parameter.....	35
5.1.2. Stream feature parameter	42
5.2. Function class definition	43
5.2.1. Feature	43
5.2.2. IntFeature	45
5.2.3. FloatFeature	48
5.2.4. EnumFeature	51
5.2.5. BoolFeature	53
5.2.6. StringFeature	54
5.2.7. BufferFeature	56
5.2.8. CommandFeature.....	57
5.3. Data type definition.....	58
5.3.1. DeviceClass.....	58
5.3.2. AccessStatus	58
5.3.3. AccessMode	58
5.3.4. FrameStatus	59
5.3.5. PixelFormatEntry	59
5.3.6. PixelSizeEntry.....	60
5.3.7. PixelColorFilterEntry	60
5.3.8. AcquisitionModeEntry	60
5.3.9. TriggerSourceEntry.....	60
5.3.10. TriggerActivationEntry	61
5.3.11. ExposureModeEntry	61
5.3.12. UserOutputSelectorEntry	61
5.3.13. UserOutputModeEntry.....	61
5.3.14. GainSelectorEntry	61
5.3.15. BlackLevelSelectEntry.....	61
5.3.16. BalanceRatioSelectorEntry	62
5.3.17. AALightEnvironmentEntry	62
5.3.18. UserSetEntry	62
5.3.19. AWBLampHouseEntry.....	62
5.3.20. TestPatternEntry	62
5.3.21. TriggerSelectorEntry	63
5.3.22. LineSelectorEntry	63
5.3.23. LineModeEntry.....	63
5.3.24. LineSourceEntry	63
5.3.25. LutSelectorEntry	64
5.3.26. TransferControlModeEntry	64
5.3.27. TransferOperationModeEntry	64
5.3.28. TestPatternGeneratorSelectorEntry	64
5.3.29. ChunkSelectorEntry.....	64
5.3.30. BinningHorizontalModeEntry.....	64
5.3.31. BinningVerticalModeEntry	64

5.3.32. AcquisitionStatusSelectorEntry	64
5.3.33. GammaModeEntry	65
5.3.34. ColorTransformationModeEntry	65
5.3.35. ColorTransformationValueSelectorEntry	65
5.3.36. AutoEntry	65
5.3.37. SwitchEntry	65
5.3.38. RegionSendModeEntry	66
5.3.39. RegionSelectorEntry	66
5.3.40. BayerConvertType	66
5.3.41. ValidBit	66
5.3.42. ImageMirrorMode	66
5.3.43. ActualBits	67
5.3.44. TimerSelectorEntry	67
5.3.45. TimerTriggerSourceEntry	67
5.3.46. CounterSelectorEntry	67
5.3.47. CounterEventSourceEntry	67
5.3.48. CounterResetSourceEntry	67
5.3.49. CounterResetActivationEntry	67
5.4. Interface definition	68
5.4.1. DeviceManager	68
5.4.2. Device	74
5.4.3. DataStream	77
5.4.4. RawImage	81
5.4.5. RGBImage	87
5.4.6. ARGBImage	90
5.4.7. ByteBuffer	94
5.4.8. Utility	96
6. Revision History	98

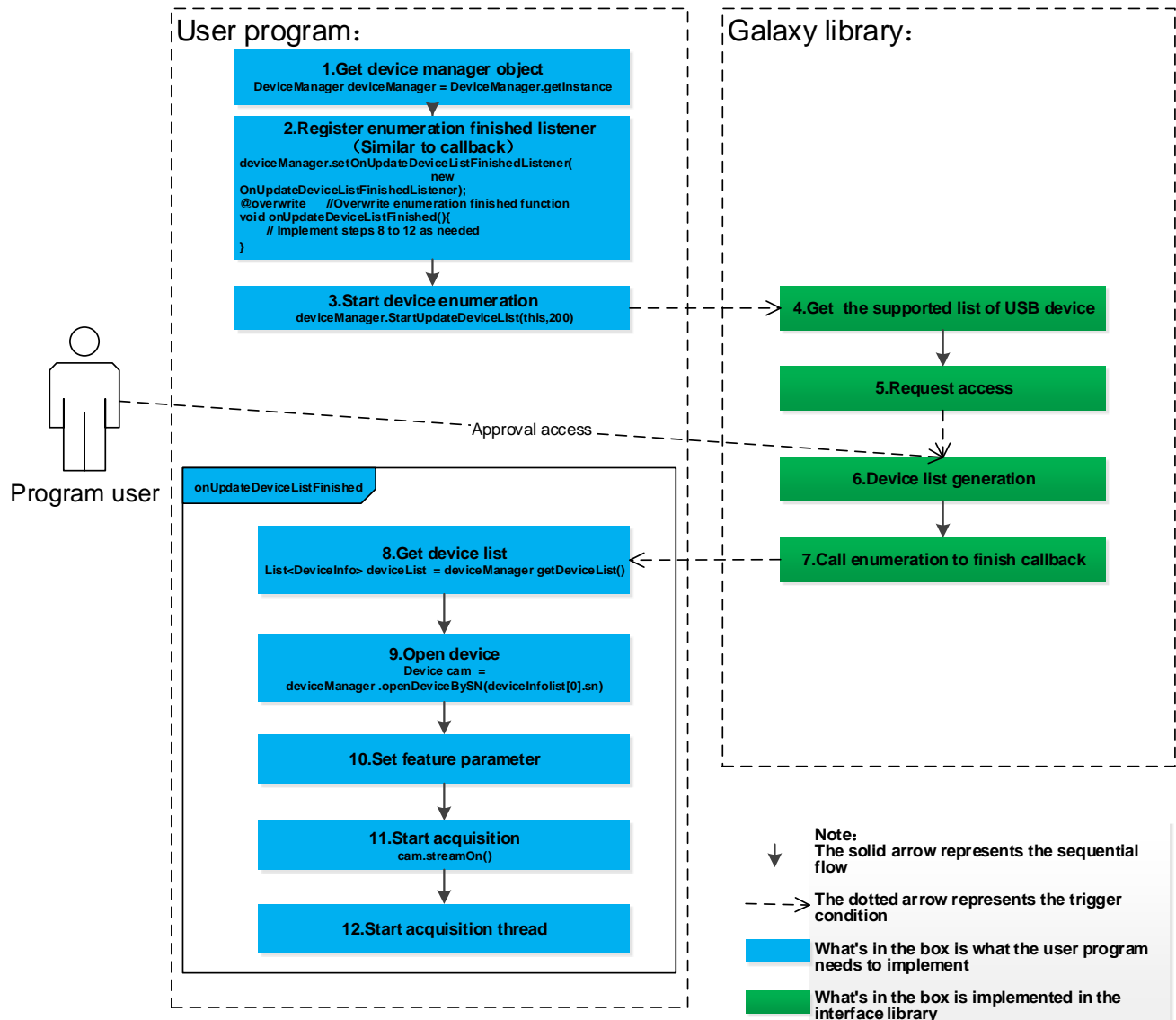
1. Camera Workflow

1.1. Overall workflow

1.1.1. General enumeration method



1.1.2. Enumeration without Root permission



Note:

- Steps 8~12 of the figure demonstrate the operations that can be performed in the override completion response function after the enumeration is completed, but not limited to these operations.
- This enumeration method cannot be used when the Android system not connects a U3V camera.
- Starting enumeration without Root permission simultaneously in multiple threads is not supported.

1.1.3. Selection of enumeration methods

➤ Non-Android system

Use the general enumeration method.

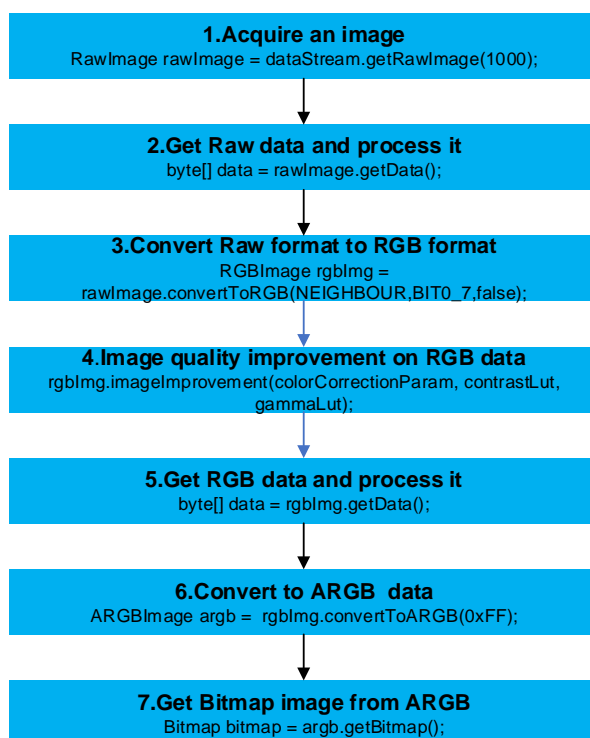
➤ Android system

- 1) **Enumeration without Root permission** method should be used under the general Android system, since the general applications do not have permission to access USB devices.
- 2) **General enumeration** method should be used under the customized Android system, since the applications have permission to access USB devices.
- 3) **General enumeration** method should be used when the user only using the non-U3V cameras.

1.2. Image acquisition flow

1.2.1. getRawImage method

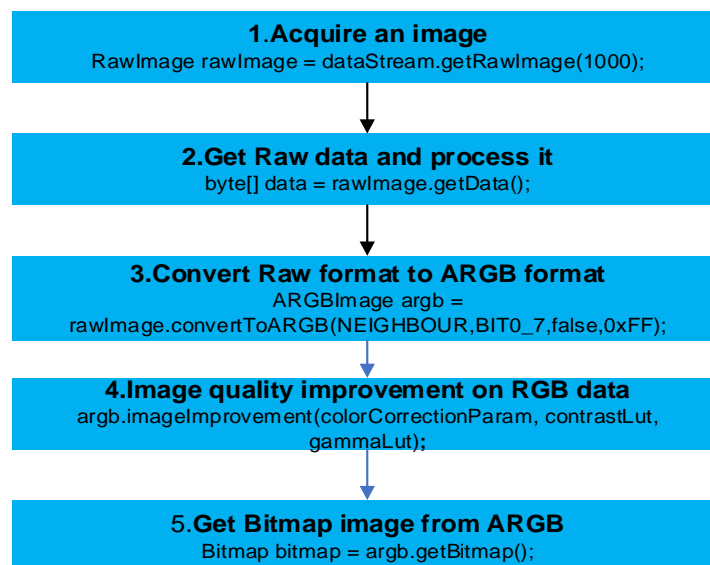
➤ Optional flow 1



Note:

1. Steps 2~7 of the flow are optional, which are implemented in the program according to requirements.
2. Image processing requirements can be met:
 - 1) Raw image data can be processed and saved.
 - 2) Color correction, Gamma, contrast, etc. can be performed on RGB data.
 - 3) Other processing for RGB data can be performed as needed, and each pixel is saved in 3 bytes in the order of R->G->B (corresponding to step 5).
 - 4) Bitmap format data can be got, and can be processed and displayed as needed.

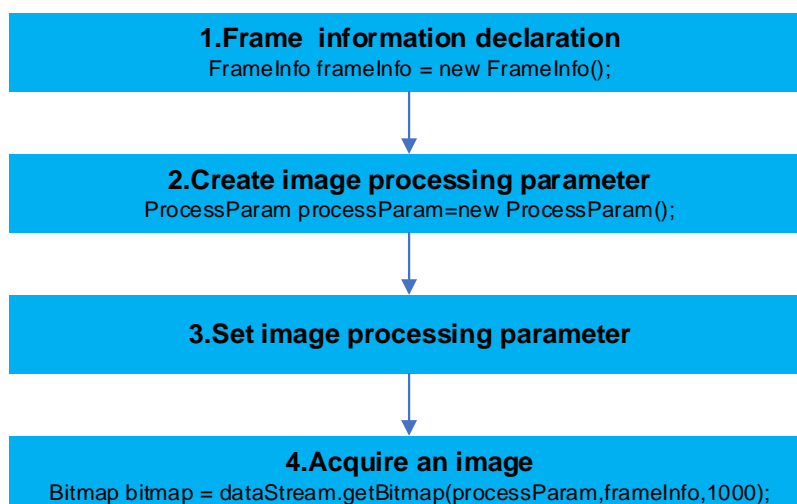
➤ **Optional flow 2**



Note:

1. Steps 2~7 of the flow are optional, which are implemented in the program according to requirements.
2. Image processing requirements that can be met:
 - 1) Raw image data can be processed and saved.
 - 2) Color correction, Gamma, contrast, etc. can be processed (corresponding to step 4).
 - 3) Bitmap format data can be got, and can be processed and displayed as needed.

1.2.2. getBitmap method

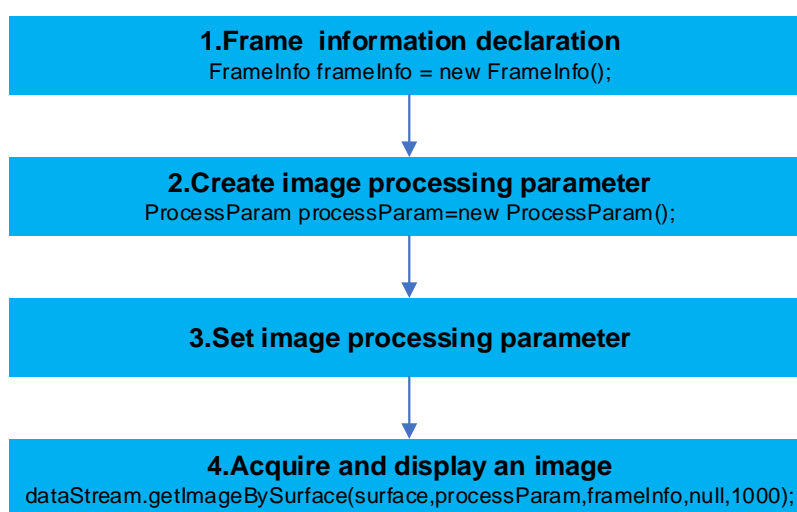


Note:

1. It is recommended to execute the 1~2 steps of the flow only once when starting the acquisition.

2. Step 3 is optional. If not executed, the default image acquisition and processing parameter and the default processing parameter are used (**Note: Color correction, Gamma, contrast, etc. are not processed by default**).
3. Image processing function:
 - 1) Color correction, Gamma, contrast, etc.
 - 2) Bitmap format data can be got, and can be processed and displayed as need.
4. Advantage:
 - 1) Zero-copy, high image acquisition efficiency.
 - 2) Calling is simple.

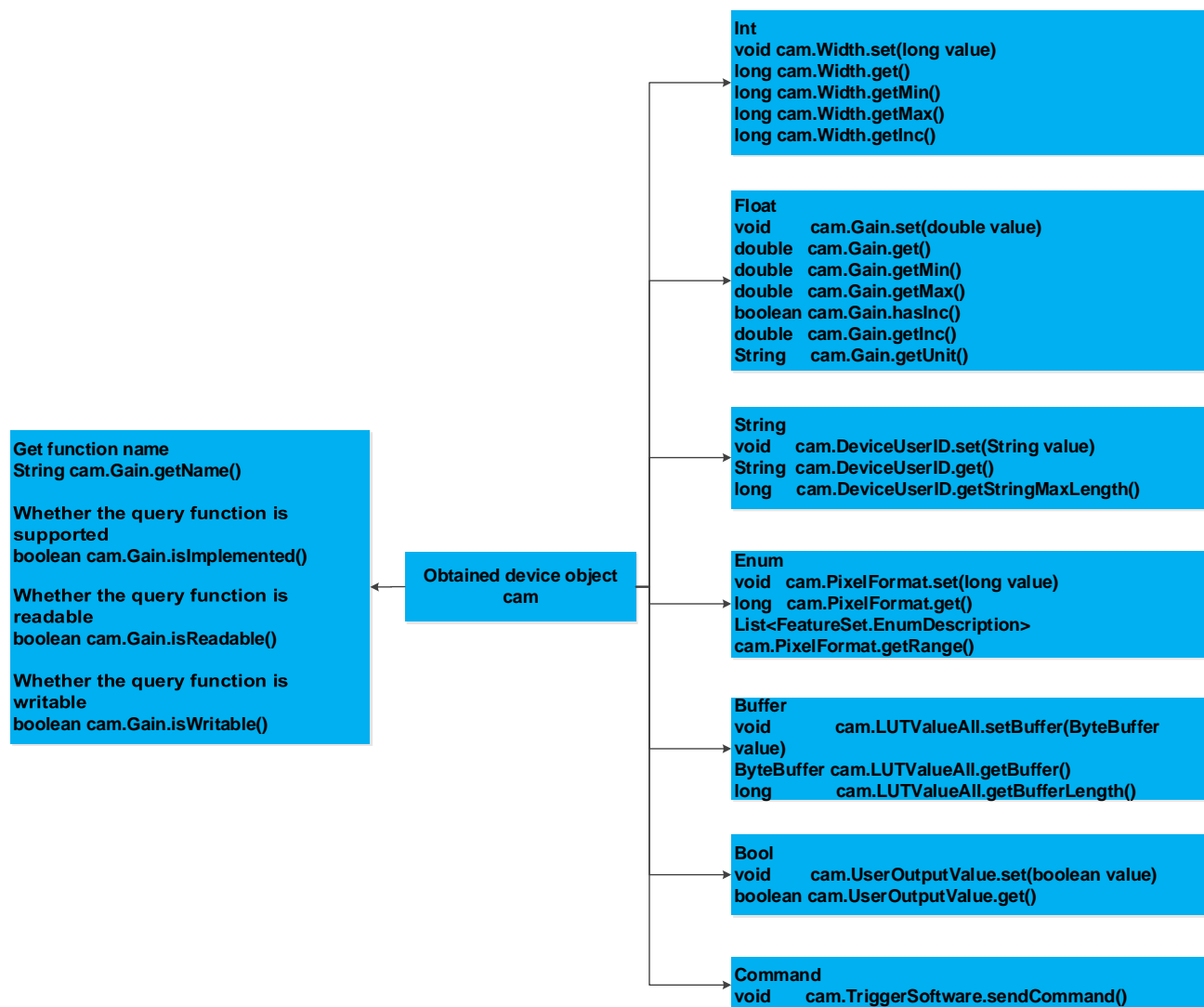
1.2.3. getImageBySurface method



Note:

1. It is recommended to execute the 1~2 steps of the flow only once when starting the acquisition.
2. Step 3 is optional. If not executed, the default image acquisition and processing parameter and the default processing parameter are used (**Note: Color correction, Gamma, contrast, etc. are not processed by default**).
3. Image processing function:
 - 1) Color correction, Gamma, contrast, etc.
 - 2) Display on the specified interface widget.
4. Advantage:
 - 1) Zero-copy, high image acquisition efficiency.
 - 2) Calling is simple.

1.3. Function control flow



1.4. Overall code sample

1.4.1. General enumeration method

```

// Import device library
import galaxy.Device;
import galaxy.DeviceManager;

// Get the instance of device manager
DeviceManager deviceManager= DeviceManager.getInstance();
// Enumerate the device, the parameter is timeout time. Return the list of
// enumerated device
List<DeviceInfo> devInfoList = deviceManager.updateDeviceList(200);
if (devInfoList.size() == 0)
{
    // Prompt that no device is enumerated
    return;
}
  
```

```
}
// Open the first device
// Open the device by index, and input 1, 2, 3..., note that the index starts
// at 1
Device cam = deviceManager.openDeviceByIndex(1);
// Get the object of 0th stream channel. The currently supported cameras
// only have one channel
DataStream dataStream = cam.getDataStream(0);

// Set parameters as needed...`
// Start acquisition
cam.streamOn();

// Loop acquire and process image data
for(int i = 0; i < 10; ++i){
    RawImage rawImage = null;
    try{
        // Get an image, the parameter is timeout time, and its unit is ms
        rawImage = dataStream.getRawImage(500);
    }
    catch (ExceptionSet.Timeout timeout) {
        // If the timeout is captured, please continue directly
        continue;
    }
    catch (ExceptionSet exceptionSet) {
        // If other exception is captured, please continue trying to get the
        // image or exit the loop after printing the exception message
        exceptionSet.printStackTrace();
        // continue;
    }
    // Image processing...
}

// Stop acquisition
cam.streamOff();
// Close device
cam.closeDevice();
```

1.4.2. Enumeration without Root permission

```
// Import device library
import galaxy.Device;
import galaxy.DeviceManager;

// Get the instance of device manager
DeviceManager deviceManager= DeviceManager.getInstance();
// Override the enumeration finished function
deviceManager.setOnUpdateDeviceListFinishedListener(new
DeviceManager.OnUpdateDeviceListFinishedListener() {
    @Override
    public void onUpdateDeviceListFinished(){
        List<DeviceInfo> deviceInfoList = deviceManager.getDeviceInfoList();
        if(deviceInfoList.size() == 0){
            // Prompt that no device is enumerated (or no device is connected)
        }
        else{
            // Perform the operations such as open device, start acquisition,
            // etc. Their code is the same as the code in the "General
            // enumeration method"
        }
    }
}
```

```
});  
// Start enumeration, the first parameter context can input this only in  
// MainActivity  
deviceManager.startUpdateDeviceList(this, 200)
```

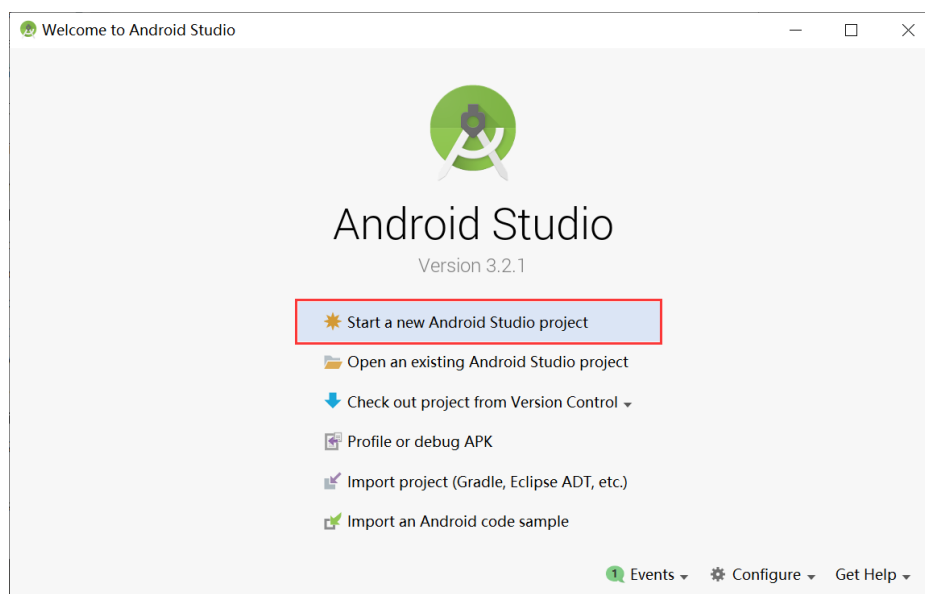
2. Build Environment

2.1. Android

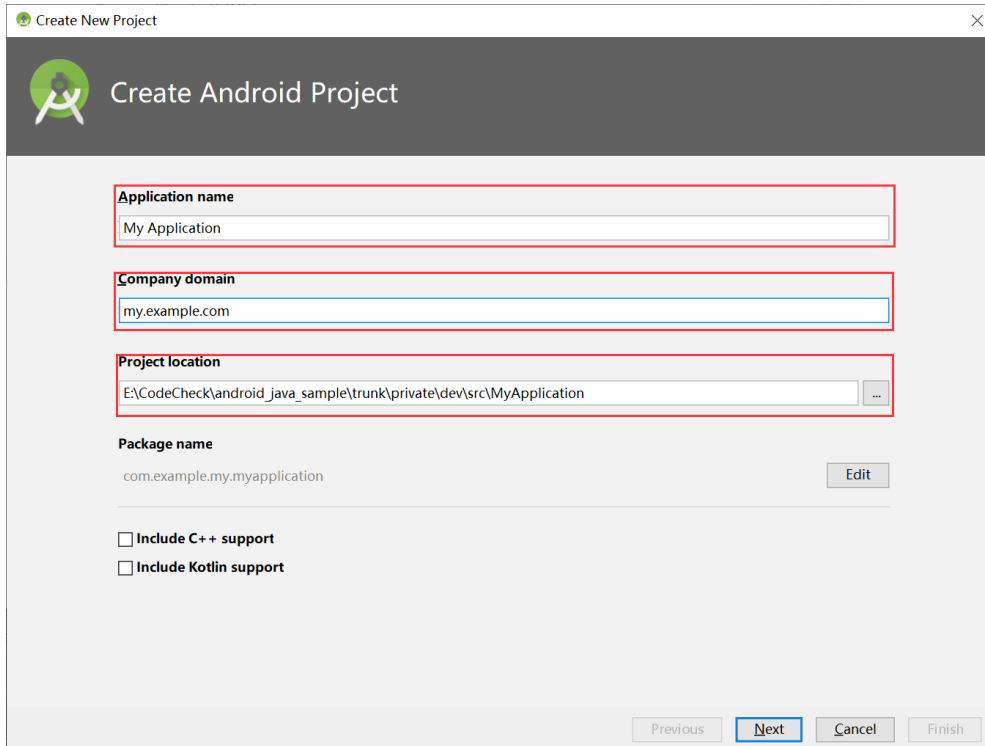
This document takes Android Studio 3.2.1 as an example to introduce the process of building development environment.

2.1.1. Create project

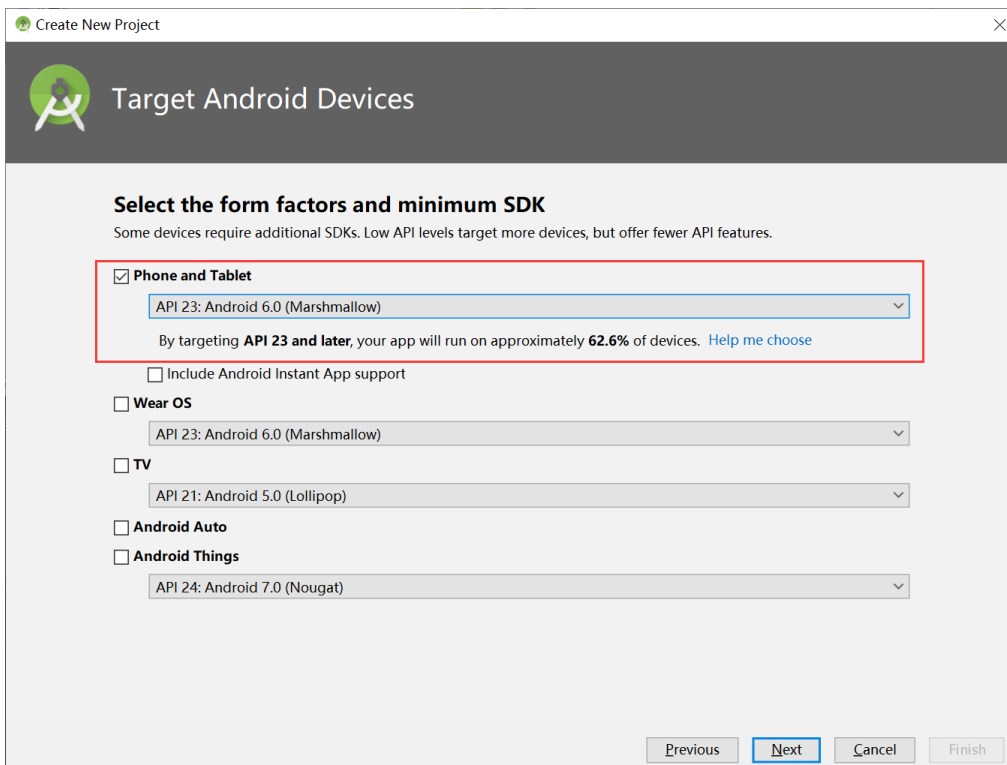
1) Select "start a new Android Studio project" in a new Android Studio project page.



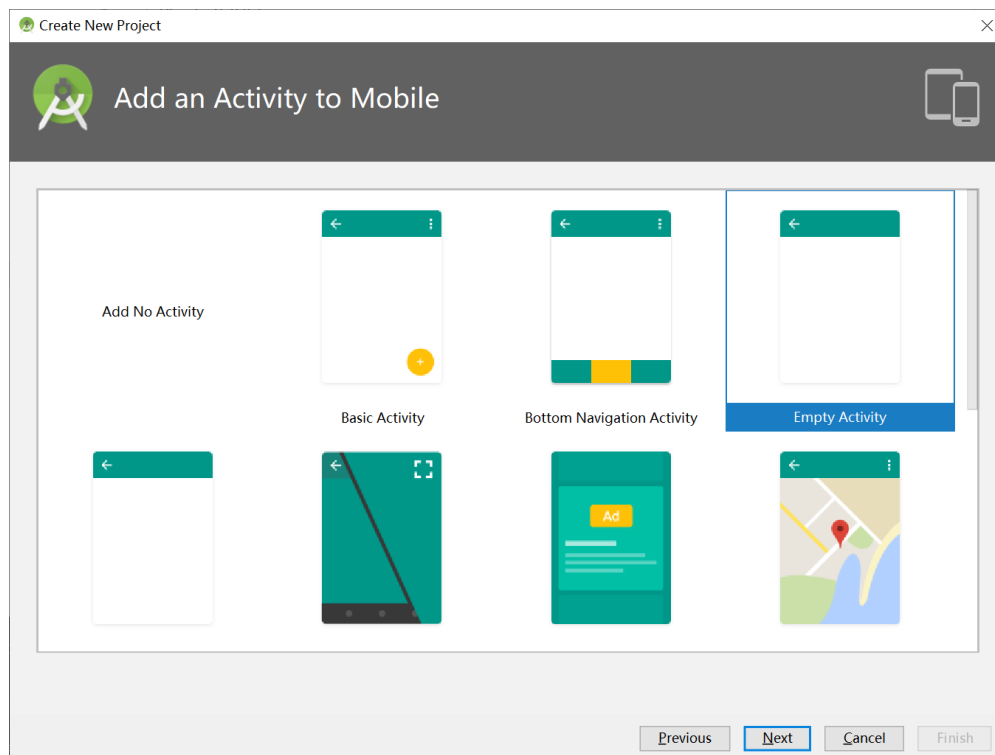
2) Fill in the pop-up page like this: fill "My Application" in the Application name, set the project path in the Project location. Others remains the default. Select "Next" to continue after completing the fill.



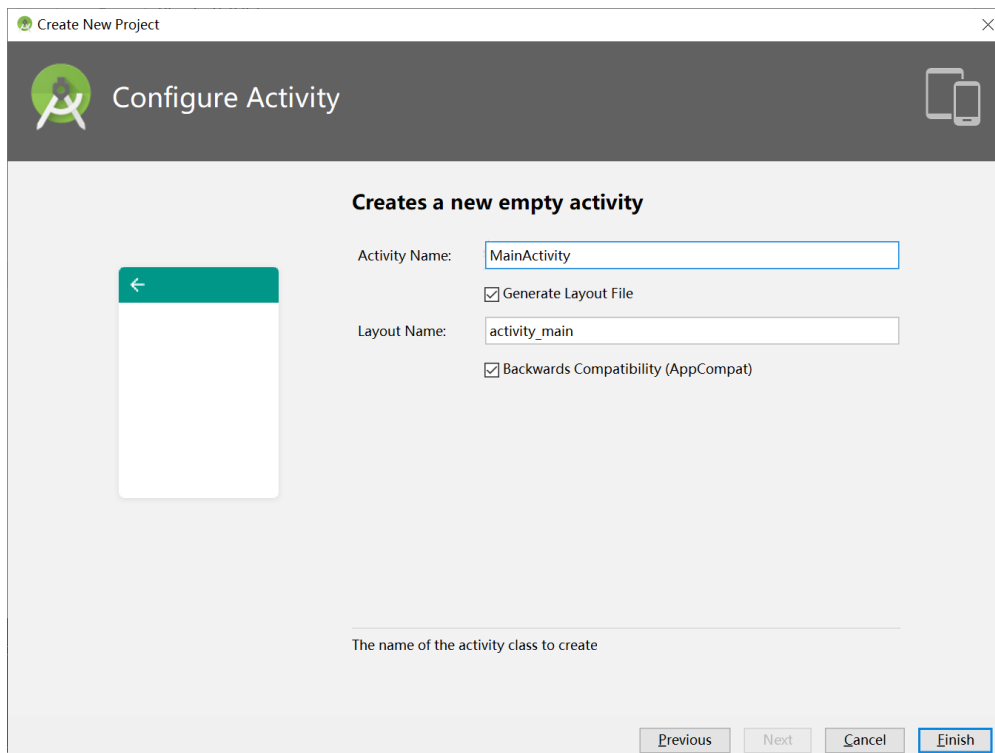
- 3) Then, the platform selection page is popped up. Currently SDK only supports Phone and Tablet platform, and Android Things platform. Android API supports Android 6.0 or above. Please select the platform and API version that meet the requirements. Select "Next" to continue after completing the correct selection.



- 4) Select a required Activity template in the "Add an Activity to Mobile" page. Use the default option Empty Activity here. Select "Next" to continue after you have determined the template.

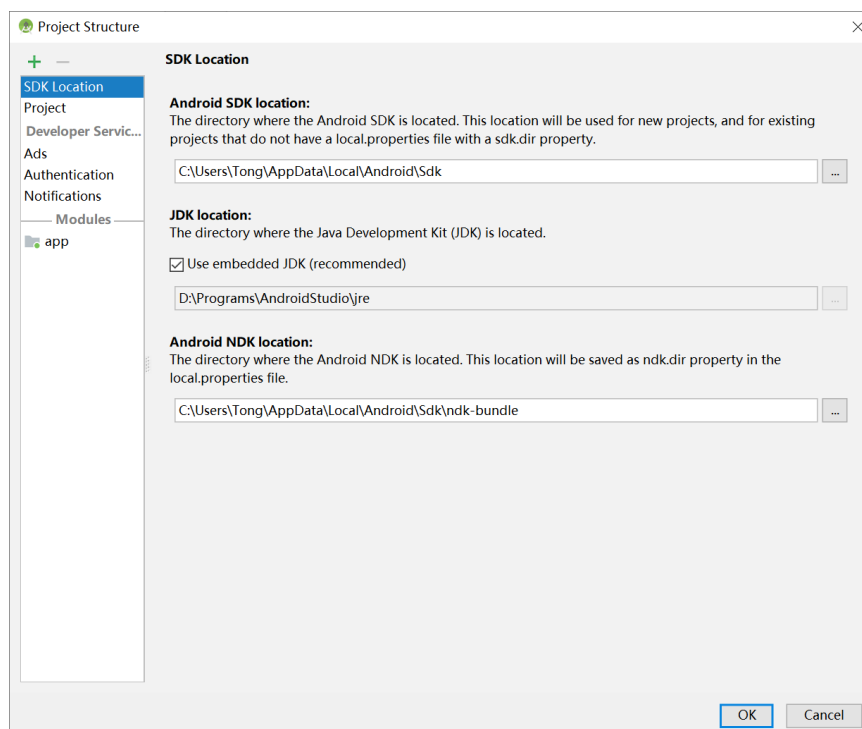


- 5) Next you can modify the name of the Activity and Layout in the "Configure Activity" page. Use the default option here. Select "Finish" to finish the project creation.

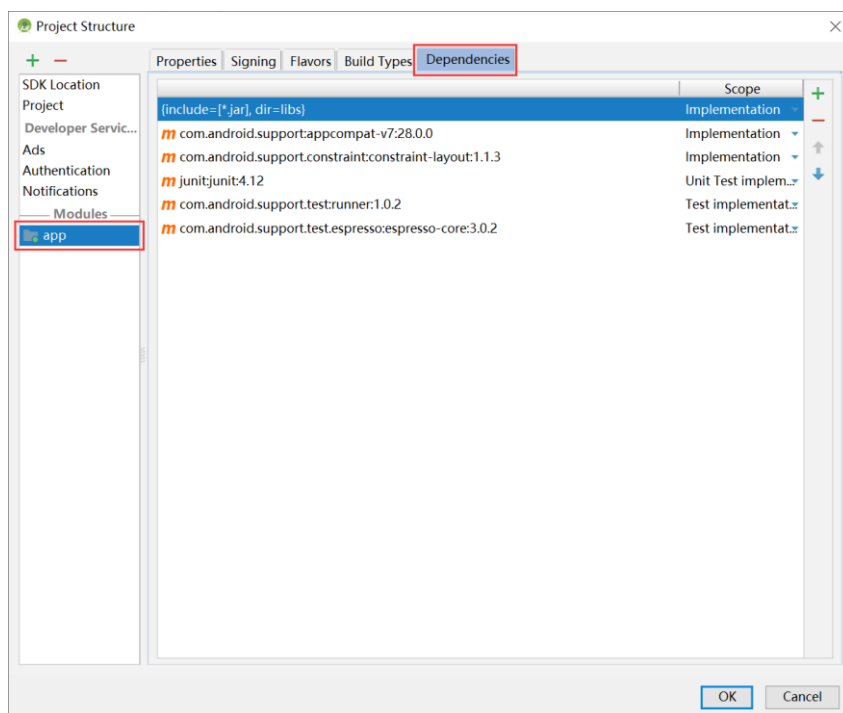


2.1.2. Importing library

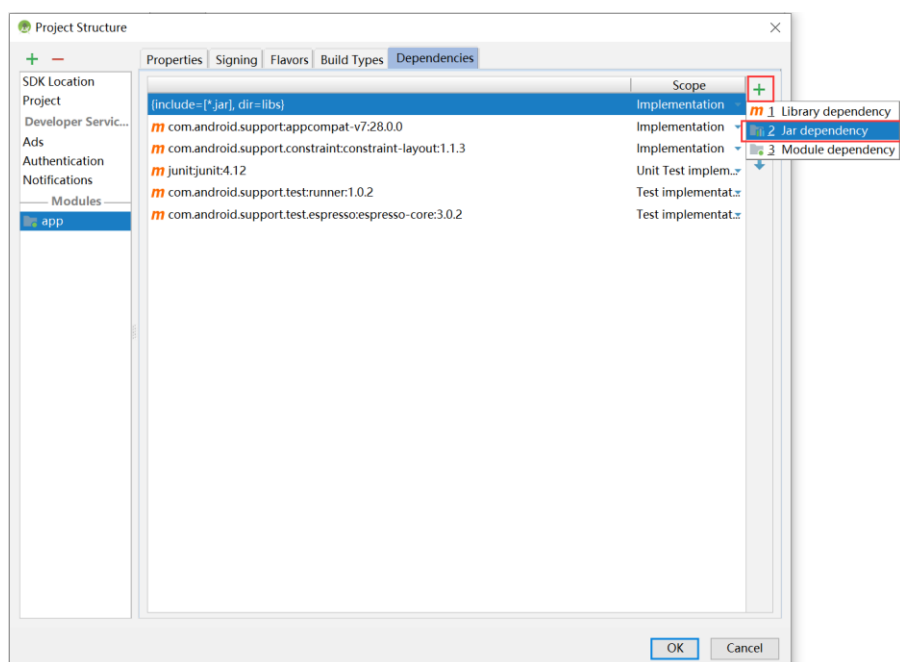
- 1) Copy the "galaxy-android-source.jar" file under the "\lib\galaxy-android-source\" directory of the SDK package to the library directory ("MyApplication\app\libs") under the project directory.
- 2) Copy the "jniLibs" directory under the "\lib\galaxy-android-jniLibs" directory of the SDK package to the main directory ("MyApplication\app\src\main\") under the project directory.
- 3) After the copy of the library is finished, enter the project structure setting page by selecting "File->Project Structure..." successively, or using shortcuts "Ctrl + Alt + Shift + s". As shown in the figure below:



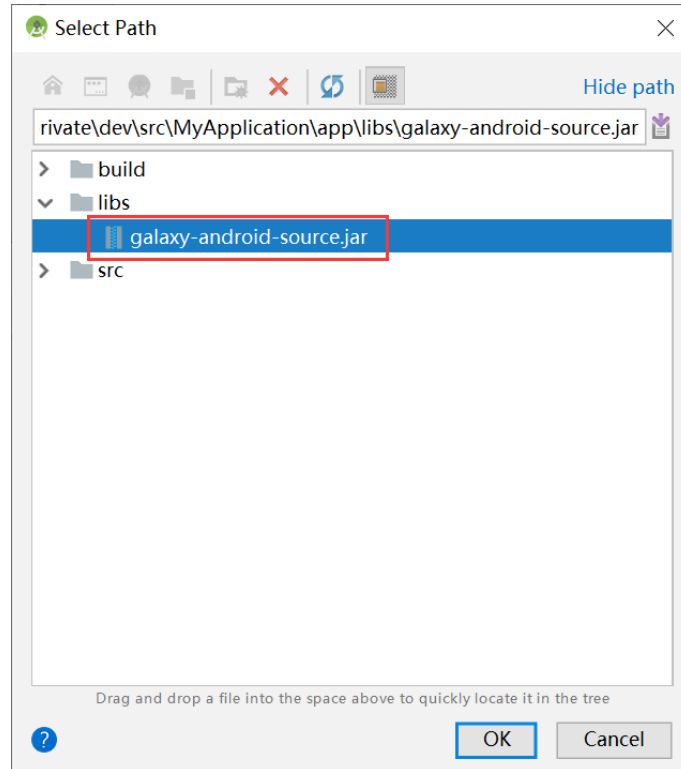
- 4) Select "app->Dependencies" successively in project structure configuring page, the page shown below will appear:



5) Select the plus sign (+) on the right side of the page and 3 options will appear, and select "2 Jar dependency".



6) After selecting, a dialog box will pop up:



- 7) Select the "galaxy-android-source.jar" package under the libs directory, then select "OK" to finish the selection.
- 8) Continue to select "OK" in the "Project Structure" page to finish importing library.

2.1.3. javadoc configuring

The javadoc is to import annotations of the interface, which ensures that the annotations for each interface can be seen when the code is written, and then configure it as needed. The configuration process is as follows:

- 1) Copy the "galaxy-android-source.jar" file under the "\lib\galaxy-android-source\" directory of the SDK package to the library directory ("MyApplication\app\libs") under the project directory.
- 2) Find the xml file corresponding to the imported "galaxy-android-source.jar" file in the directory of the project window ("MyApplication\idea\libraries"). As shown in the figure below:



3) Open the xml file and change the tag of "<JAVADOC />" to:

```
<JAVADOC>
    <root url="jar://$PROJECT_DIR$/app/libs/galaxy-android-javadoc.jar!/" />
</JAVADOC>
```

After the modification is finished, as shown in the figure below:



3. Programming Guide

3.1. Importing library

The libraries should be imported at the beginning of the program in order to use the interface libraries.

Sample code:

```
// Import device manager class from the device library
import galaxy.DeviceManager;

// Get the instance of device manager
DeviceManager deviceManager = DeviceManager.getInstance();
```

3.2. Enumeration device

The user can get the device information list `List<DeviceInfo> devInfoList` by calling enumeration interface to enumerate all currently available devices. The number of elements in the device information list is the number of devices enumerated, the data type of the elements in the list is `DeviceInfo`. The device information saved in `DeviceInfo` are shown in the table below:

Information	Got method	Type
Device index	<code>getIndex()</code>	int
Vendor name	<code>getVendorName()</code>	string
Model name	<code>getModelName()</code>	string
Device serial number	<code>getSN()</code>	string
Device display name	<code>getDisplayName()</code>	string
Device identification	<code>getDeviceID()</code>	string
User-defined name	<code>getUserID()</code>	string
Access status	<code>getAccessStatus()</code>	AccessStatus
Device class	<code>getDeviceClass()</code>	DeviceClass
MAC address (GEV camera only)	<code>getMAC()</code>	string
IP address (GEV camera only)	<code>getIP()</code>	string
Subnet mask (GEV camera only)	<code>getSubnetMask()</code>	string
Gateway (GEV camera only)	<code>getGateway()</code>	string
Nic MAC address of the host to which the camera is connected (GEV camera only)	<code>getNICMAC()</code>	string
Nic IP address of the host to which the camera is connected (GEV camera only)	<code>getNICIP()</code>	string
Nic subnet mask of the host to which the camera is connected (GEV camera only)	<code>getNICSubnetMask()</code>	string
Nic gateway of the host to which the camera is connected (GEV camera only)	<code>getNICGateway()</code>	string
Nic description of the host to which the camera is connected (GEV camera only)	<code>getNICDescription()</code>	string

3.2.1. General method

Sample code:

```
// Get the instance of device manager
DeviceManager deviceManager= DeviceManager.getInstance();

// Enumerate device (The enumeration timeout time is 200ms)
List<DeviceInfo> devInfoList = deviceManager.updateDeviceList(200);
if (devInfoList.size() == 0)
{
    // Prompt that no device is enumerated
}
else
{
    // Perform operations such as opening device, starting acquisition, etc.
    // after enumerating the device successfully
}
```

Note:

In addition to the enumeration interface [updateDeviceList](#) above, [DeviceManager](#) provides another general enumeration interface [updateAllDeviceList](#):

- 1) For non-GEV cameras, the two enumeration interfaces are functionally identical.
- 2) For GEV cameras, the enumeration mechanisms are different:

updateAllDeviceList: Using entire network enumeration, you can enumerate all GEV cameras on the LAN.

updateDeviceList: Using subnet enumeration, you can only enumerate GEV cameras that are on the same network segment within the LAN.

3.2.2. No Root permission method

Please see section [1.4.2](#) for sample code.

Note:

In addition to the start enumeration interface [startUpdateDeviceList](#) of the sample code in section [1.4.2](#), [DeviceManager](#) provides another start enumeration interface [startUpdateAllDeviceList](#):

- 1) For non-GEV cameras, the two enumeration interfaces are functionally identical.
- 2) For GEV cameras, the enumeration mechanisms are different:

startUpdateAllDeviceList: Starting entire network enumeration process of the GEV cameras and the enumeration process of U3V cameras, you can enumerate all GEV cameras and U3V cameras in the LAN.

startUpdateDeviceList: Starting subnet network enumeration process of the GEV cameras and the enumeration process of U3V cameras, you can only enumerate GEV cameras and U3V cameras that are on the same network segment within the LAN.

- 3) General enumeration method **must** be used when only using the **non-U3V** cameras.
- 4) **Starting the process of no root permission enumeration simultaneously in multiple threads is not supported.**

3.3. Open the device

The user can open the device by the following five different methods:

- 1) Device [openDeviceBySN](#)(String strSN, [AccessMode](#) accessMode).
- 2) Device [openDeviceByUserID](#) (String strUserID, [AccessMode](#) accessMode).
- 3) Device [openDeviceByIndex](#)(int index, [AccessMode](#) accessMode).
- 4) Device [openDeviceByIP](#)(String strIP, [AccessMode](#) accessMode).
- 5) Device [openDeviceByMAC](#)(String strMAC, [AccessMode](#) accessMode).

Parameters:

strSN	The device serial number
strUserID	The user-defined name
index	The device index (1, 2, 3...)
strIP	The device IP address (non- GEV cameras are not support)
strMAC	The device MAC address (non- GEV cameras are not support)

Note:

The last two functions are only for GEV cameras.

The user can call the [closeDevice](#) interface provided by [Device](#) to close the device, and release all device resources.

Sample code:

```
// Open device
// Method 1
// Open the device by serial number, or you can input the string of SN, for
// example "NC0150120001"
String strSN = devInfoList.get(0).getSN();
Device cam = deviceManager.openDeviceBySN(strSN);

// Method 2
// Open the device by user ID
// String strUserID = devInfoList.get(0).getUserID();
// Device cam = deviceManager.openDeviceByUserID(strUserID);

// Method 3
// Open the device by index, and input 1, 2, 3... Note that the index starts
// at 1
// int index = devInfoList.get(0).getIndex();
// Device cam = deviceManager.openDeviceByIndex(index);

// The following opening methods are only for GEV cameras
// Method 4
// Open the device by IP address
```

```
// String strIP= devInfoList.get(0).getIP() ;
// Device cam = deviceManager.openDeviceByIP(strIP,
//                                     EnumDefineSet.AccessMode.CONTROL);

// Method 5
// Open the device by MAC address
// String strMAC= devInfoList.get(0).getMAC()
// Device cam = deviceManager.openDeviceByMAC(strMAC,
//                                     EnumDefineSet.AccessMode.CONTROL);

// Close device
cam.closeDevice();
```

3.4. Acquisition control

The user can call [Device.streamOn\(\)](#) and [Device.streamOff\(\)](#) to start and stop acquisition after the device is opened successfully and the camera acquisition parameters are set.

You can get the number of device stream channels through the [Device.getStreamChannelNum\(\)](#) interface. All cameras supported by the current library **only have one stream channel**.

The first stream channel [DataStream](#) class object can be got by [Device.getDataStream\(0\)](#).

Sample code:

```
// Get DataStream class object
DataStream dataStream = cam.getDataStream(0);

// Set the number of images acquisition buffer before starting the
// acquisition. The default number is 5
dataStream.setAcquisitionBufferNumber(10);

// Start acquisition
cam.streamOn();

// Image acquisition... See section 3.5 for details

// If you need to use the getRawImage or getBitmap to call the latest
// acquired image every time, please call dataStream.flushQueue() first
// before calling them

// Stop acquisition
cam.streamOff();
```

3.5. Image acquisition and processing

The [DataStream](#) class provides three interfaces ([getRawImage](#)/[getBitmap](#)/[getImageBySurface](#)) for acquiring images. Users can select one of them according to the needs. **Note that they cannot be used at the same time.**

3.5.1. getRawImage method

➤ Raw image acquisition

Get Raw image using the [getRawImage](#) interface. And the return value is the object of the [RawImage](#) class.

Sample code:

```
RawImage rawImage = null;
try{
    // The parameter is timeout time, and its unit is ms
    rawImage = dataStream.getRawImage(500);
}
catch(ExceptionSet.Timeout timeout) {
    // Timeout is captured. If the timeout is captured during loop acquisition,
    // continue directly without printing error message
    continue;
}
catch(ExceptionSet exceptionSet) {
    // Other exception is captured. If the exception is captured during loop
    // acquisition, continue trying to get the image or stop acquisition,
    // after printing the exception message
    exceptionSet.printStackTrace();
    // continue;
}

if(rawImage != null){
    // Get Raw image data
    byte[] rawData = rawImage.getData();

    // Process image data as needed

    // Save Raw image
    rawImage.save("Img0.raw");
}
```

➤ Convert Raw images to RGB images

Convert Raw format images to RGB format images by calling the [convertToRGB](#) interface. And return the [RGBImage](#) class object.

Sample code:

```
// Convert Raw format images to RGB format images
RGBImage rgbImage = rawImage.convertToRGB(
    BayerConvertType.NEIGHBOUR, // Neighbourhood method Bayer
                                // conversion
    ValidBit.BIT0_7),           // The image format is not 8
                                // bits, take 0~7 bits
    false);                     // false means don't reverse

// Improve image quality based on ARGB data. And users can choose whether to
// process according to their needs
// Get the image quality improvement parameter: Gamma
ByteBuffer gammaLut = null;
if(cam.GammaParam.isReadable()){
    double gammaValue = cam.GammaParam.get();
    gammaLut = Utility.getGammaLut(gammaValue);
}
```

```
}
else{
    // When the camera does not support the recommended Gamma value, there
    // are two methods to deal with it:
    // 1.gammaLut = null;This method does not perform Gamma adjustment

    // 2.Given a Gamma value as required,then convert it to gammaLut
}

// Get the image quality improvement parameter: Contrast
ByteBuffer contrastLut = null;
if(cam.ContrastParam.isReadable()){
    long contrastValue = cam.ContrastParam.get();
    contrastLut = Utility.getContrastLut(contrastValue);
}
else{
    // When the camera does not support the recommended Contrast value, the
    // following two methods can be used:
    // 1. contrastLut = null;Contrast adjustment is not performed in this
    // method
    // 2. Given a Contrast value as needed, then convert it to contrastLut
}

// Get color correction parameter
long colorCorrectionParam = cam.ColorCorrectionParam.get();

// Achieve image quality improvement
rgbImage.imageImprovement(colorCorrectionParam, contrastLut, gammaLut);

// Get RGB format image data
byte[] rgbData = rgbImage.getData();

// Process rgbData data as needed ...

// Convert RGB data to ARGB data, then get the Bitmap
ARGBImage argbImage = rgbImage.convertToARGB((byte) 0xFF); // The parameter
// is the channel value of Alpha

// Get Bitmap from ARGBImage object
Bitmap bitmap = argbImage.getBitmap();
```

➤ Convert Raw images to ARGB images

Convert Raw format images to ARGB format images by calling the [convertToARGB](#) interface. And return the [ARGBImage](#) class object.

Sample code:

```
// Convert Raw format images to ARGB format images
ARGBImage argbImage = rawImage.convertToARGB(
    BayerConvertType.NEIGHBOUR, // Neighbourhood method Bayer
                                // conversion
    ValidBit.BIT0_7),          // The image format is not 8
                                // bits, take 0~7 bits
    false,                     // false means don't Reverse
    (byte) 0xFF);              // Alpha channel value is 0xFF

// Improve image quality based on ARGB data. And users can choose whether to
// perform it according to the needs
// Get the image quality improvement parameter: Gamma
```

```
ByteBuffer gammaLut = null;
if (cam.GammaParam.isReadable()) {
    double gammaValue = cam.GammaParam.get();
    gammaLut = Utility.getGammaLut(gammaValue);
}
else {
    // When the camera does not support the recommended Gamma value, the
    // following two methods can be used:
    // 1.gammaLut = null;Gamma adjustment is not performed in this method

    // 2.Given a Gamma value as required,then convert it to gammaLut
}

// Get the image quality improvement parameter: Contrast
ByteBuffer contrastLut = null;
if (cam.ContrastParam.isReadable()) {
    long contrastValue = cam.ContrastParam.get();
    contrastLut = Utility.getContrastLut(contrastValue);
}
else {
    // When the camera does not support getting the recommended Contrast
    // value, the following two methods can be used:
    // 1.contrastLut = null;Contrast adjustment is not performed in this
    // method

    // 2.Given a Contrast value as needed,then convert it to contrastLut
}

// Get color correction parameter
long colorCorrectionParam = cam.ColorCorrectionParam.get();

// Achieve image quality improvement
argbImage.imageImprovement(colorCorrectionParam, contrastLut, gammaLut);

// Get ARGB format image data
int[] argbData = argbImage.getData();

// Process argbData data as needed ...

// Get Bitmap from ARGBImage object
Bitmap bitmap = argbImage.getBitmap();
```

3.5.2. getBitmap method

[getBitmap](#) is an acquisition method provided by [DataStream](#) class, which mainly converts a Raw image that acquired from the camera to a Bitmap to output. The return value is Bitmap type.

The first parameter type is processParam, mainly includes related parameters of image conversion and image quality improvement. As shown in the following table:

Parameter name	Parameter type	Default	Parameter description
bayerConvertType	BayerConvertType	NEIGHBOUR	Convert Bayer format image to RGB format image. The default method is neighbour method
validBit	ValidBit	BIT0_7	Valid bit selection that converting a non-8-bit Raw image to 8-bit Raw image

flip	boolean	false	Whether to flip
mirror	boolean	false	Whether to perform image mirror
imageMirrorMode	ImageMirrorMode	HORIZONTAL_MIRROR	Mirror mode. The parameter takes effect when mirror is set to true.
colorCorrectionParam	long	0	Color correction parameter, 0 means no processing
gammaLut	ByteBuffer	null	Gamma LUT, null means no processing
contrastLut	ByteBuffer	null	Contrast LUT, null means no processing
alpha	Byte	0xFF	Alpha channel value

The second parameter is the output parameter of FrameInfo type, mainly contains the basic information of the acquired image. As shown in the following table:

Information name	Type	Description
status	int	Image status, its status meaning refers to FrameStatus , getBitmap interface returns null when status value is not 0.
width	int	Image width
height	int	Image height
offsetX	int	Get image OffsetX
offsetY	int	Get image OffsetY
frameID	long	Frame ID
timestamp	long	Timestamp

The third parameter is timeout time, it is the maximum blocking time when calling this interface. The unit is ms.

Sample code:

```

Bitmap bitmap = null;
ProcessParam processParam = new ProcessParam();
FrameInfo frameInfo = new FrameInfo();

// The following code modifies the parameters related to image acquisition
// and processing to non-default values. Do not execute code when the
// default value can meet the needs
// processParam.setBayerConvertType(
//     EnumDefineSet.BayerConvertType.NEIGHBOUR); // Set interpolation
//                                             // algorithm to edge adaptive
// processParam.setValidBit(
//     EnumDefineSet.ValidBit.BIT1_8); // Valid bit selection.
//                                     // Camera PixelFormat is valid when set to non-8bits
// processParam.setFlip(true); // Flip while
//                             // interpolating
// processParam.setMirror(true); // Mirror processing
// processParam.setImageMirrorMode(
//     EnumDefineSet.ImageMirrorMode.HORIZONTAL_MIRROR); // Horizontal
//                                                         // mirror

// Initialize image quality improvement parameters (This step can be

```

```
// skipped when you don't need image quality improvement)
// Get image quality improvement parameter: Gamma
if(cam.GammaParam.isReadable()){
    double gammaValue = cam.GammaParam.get();
    ByteBuffer gammaLut = Utility.getGammaLut(gammaValue)
    processParam.setGammaLut(gammaLut);
}
else{
    // When the camera does not support the recommended Gamma value, there
    // are two methods to deal with it:
    // 1.gammaLut = null;This method does not perform Gamma adjustment

    // 2.Given a Gamma value as needed, then convert it to gammaLut
}

// Get image quality improvement parameter: Contrast
if(cam.ContrastParam.isReadable()){
    long contrastValue = cam.ContrastParam.get();
    ByteBuffer contrastLut = Utility.getContrastLut(contrastValue)
    processParam.setContrastLut(contrastLut);
}
else{
    // When the camera does not support the recommended Contrast value,
    // there are two methods to deal with it:
    // 1.contrastLut = null;This method does not perform Contrast
    // adjustment
    // 2.Given a Contrast value as needed,then convert it to contrastLut
}

// Get color correction parameter
processParam.setColorCorrectionParam(cam.ColorCorrectionParam.get());

try{
    // The parameter is timeout, and its unit is ms
    bitmap = dataStream.getBitmap(processParam, frameInfo, 500);
}
catch(ExceptionSet.Timeout timeout) {
    // Timeout is captured. If the timeout is captured during loop
    // acquisition, continue directly without printing error messages
    continue;
}
catch(ExceptionSet exceptionSet) {
    // Other exception is captured. If the exception is captured during loop
    // acquisition, continue trying to get the image or stop acquisition,
    // after printing the exception messages
    exceptionSet.printStackTrace();
    // continue;
}

if(bitmap != null && frameInfo.getStatus() == 0){
    // Process or display bitmap image as needed
}
```

3.5.3. getImageBySurface method

[getImageBySurface](#) is an acquisition method provided by [DataStream](#) class, which mainly realizes the function that acquires a Raw format image, converts and processes the image, then displays the image directly on the widget (Surface).

The first parameter type is Surface, inputs widget class object that needs to display images.

The second parameter type is ProcessParam, mainly includes related parameters of image conversion and image quality improvement. As shown in the following table:

Parameter name	Parameter type	Default	Parameter description
bayerConvertType	BayerConvertType	NEIGHBOUR	The method that converting Bayer format image to RGB format image. The default method is neighbour method
validBit	ValidBit	BIT0_7	Valid bit selection that converting non-8-bit Raw image to 8-bit Raw image
flip	boolean	false	Whether to flip
mirror	boolean	false	Whether to perform image mirror
imageMirrorMode	ImageMirrorMode	HORIZONTAL_MIRROR	Mirror mode. The parameter takes effect when mirror is set to true.
colorCorrectionParam	long	0	Color correction parameter, 0 means no processing
gammaLut	ByteBuffer	null	Gamma LUT, null means no processing
contrastLut	ByteBuffer	null	Contrast LUT, null means no processing
alpha	Byte	0xFF	Alpha channel value

The third parameter is the output parameter of FrameInfo type, mainly contains the basic information of the acquired image. As shown in the following table:

Information name	Type	Description
status	int	Image status, reference to FrameStatus for status meaning
width	int	Image width
height	int	Image height
offsetX	int	Get image OffsetX
offsetY	int	Get image OffsetY
frameID	long	Frame ID
timestamp	long	Timestamp

The fourth parameter is to save the path and name of the image. When null is inputted, the image is not saved.

The fifth parameter is timeout time which is the maximum blocking time when calling this interface. The unit is ms.

Sample code:

```
ProcessParam processParam = new ProcessParam();
FrameInfo frameInfo = new FrameInfo();
```

```
// The following code modifies the parameters related to image acquisition
// and processing to non-default values. Do not execute the code when the
// default value can meet the needs
// processParam.setBayerConvertType(
//     EnumDefineSet.BayerConvertType.NEIGHBOUR); // Set interpolation
//                                             // algorithm to edge adaptive
// processParam.setValidBit(
//     EnumDefineSet.ValidBit.BIT1_8);          // Valid bit selection,
//                                             // valid when PixelFormat is set to non-8 bits
// processParam.setFlip(true);                  // Flip while interpolating
// processParam.setMirror(true);                // Mirror processing
// processParam.setImageMirrorMode(
//     EnumDefineSet.ImageMirrorMode.HORIZONTAL_MIRROR); // Horizontal
//                                             // mirror

// Initialize image quality improvement parameters (This step can be skipped
// if image quality improvement is not required)
// Get image quality improvement parameter: Gamma
if(cam.GammaParam.isReadable()){
    double gammaValue = cam.GammaParam.get();
    ByteBuffer gammaLut = Utility.getGammaLut(gammaValue)
    processParam.setGammaLut(gammaLut);
}
else{
    // When the camera does not support the recommended Gamma value, there
    // are two methods to deal with it:
    // 1.gammaLut = null;This method does not perform Gamma adjustment

    // 2.Given a Gamma value as needed, then convert it to gammaLut
}

// Get image quality improvement parameter: Contrast
if(cam.ContrastParam.isReadable()){
    long contrastValue = cam.ContrastParam.get();
    ByteBuffer contrastLut = Utility.getContrastLut(contrastValue)
    processParam.setContrastLut(contrastLut);
}
else{
    // When the camera does not support the recommended Contrast value,
    // there are two methods to deal with it:
    // 1.contrastLut = null;This method does not perform Contrast adjustment

    // 2. Given a Contrast value as needed, then convert it to contrastLut
}

// Get color correction parameter
processParam.setColorCorrectionParam(cam.ColorCorrectionParam.get());

try{
    // Try to acquire an image and to display the image on the surface
    dataStream.getImageBySurface(
        surface,
        processParam,
        frameInfo,
        null,    // Save the path of Raw image, and the
                // image is not saved when the null is inputed
        500);    //The unit is ms
}
catch(ExceptionSet.Timeout timeout) {
    // Timeout is captured. If the timeout is captured during loop
}
```

```
// acquisition, continue directly without printing error messages
continue;
}
catch (ExceptionSet exceptionSet) {
    // Other exception is captured. If the exception is captured during loop
    // acquisition, continue trying to get the image or stop acquisition,
    //after printing the exception messages

    exceptionSet.printStackTrace();
    // continue;
}
```

3.5.4. Image quality improvement

The three kinds of image acquisition interfaces provided by the Java interface library under Android can improve the image quality. The image quality improvement function mainly includes image processing such as color correction, Gamma, contrast, etc. The user can perform the function selectively. See the sample code for the three acquisition methods above for specific usage.

1) Color correction

Device feature name: ColorCorrectionParam

Explanation: Improve the color reproduction of the camera to make the image closer to the visual perception of the human eye.

2) Gamma adjustment

Gamma value: float, range [0.1, 10.0], default value is 1

Device feature name: GammaParam

Explanation: Gamma adjustment is to make the output of the display as close as possible to the input.

3) Contrast adjustment

Contrast value: int, range [-50, 100], default value is 0

Device feature name: ContrastParam

Explanation: The brightness ratio between the bright part and the dark part of the image is called contrast. For images with high contrast, the contour of the subject is clearer and the image is clearer. Conversely, the image with low contrast has unclear contour and unclear image.

3.6. Camera control

3.6.1. Feature parameter access type

Here are three types of access to feature parameter: whether it is implemented, readable, or writable. The interfaces are designed as follows:

boolean isImplemented();	Is this current feature controller support this feature
boolean isReadable();	Is this feature readable
boolean isWritable();	Is this feature writable

It is recommended that the user query the access type of the feature before operating the feature parameter.

Sample code:

```
// Whether the function is implemented
DataStream dataStream = cam.getDataStream(0);
boolean isImplemented = dataStream.StreamTransferSize.isImplemented();
if(isImplemented)
{
    // Is this writable
    boolean isWritable = dataStream.StreamTransferSize.isWritable();
    if(isWritable)
    {
        // Set the size of URB
        dataStream.StreamTransferSize.set(512*1024);
    }

    // Is this readable
    boolean isReadable = dataStream.StreamTransferSize.isReadable();
    if(isReadable)
    {
        // Get the current size of URB
        long transferSize = dataStream.StreamTransferSize.get();
    }
}
```

3.6.2. Feature control

Two types of feature control are available: [Device feature parameter](#) and [Stream feature parameter](#).

The difference is that the control functions of the feature parameters are different:

- 1) Device feature parameter: Device information, such as width, height, exposure, gain, etc.
- 2) Stream feature parameter: Feature related to acquisition control and the acquisition of data statistics.

Interface calls are classified according to the type of feature parameters:

Int ([IntFeature](#)):

Related interfaces:

```
void set(long value);           // Set
long get();                     // Get
long getMin();                  // Get the minimum
long getMax();                  // Get the maximum
long getInc();                  // Get the step
```

Sample code:

```
// Get the maximum width of the image
long max = cam.Width.getMax();

// Set the current image width to the maximum
cam.Width.set(max);

// Get the width of current image
long value = cam.Width.get();
```

Float ([FloatFeature](#)):

Related interfaces:

```
void    set(double value);      // Set
double  get();                  // Get
double  getMin();               // Get the minimum
double  getMax();               // Get the maximum
double  getInc();               // Get the accuracy
```

Sample code:

```
// Get the maximum settable value of the exposure time
double max = cam.ExposureTime.getMax();

// Set the current exposure time to 10ms
cam.ExposureTime.set(10000);

// Get the current exposure time
double curValue = cam.ExposureTime.get();
```

Enum ([EnumFeature](#)):

Related interfaces:

```
Void          set(long enumValue); // Set
long          get();                // Get
List<EnumDescription> getRange();    // Get the list of enumeration range
```

Sample code:

```
// Get the settable range of enum value
List<EnumDescription> enumDescList = cam.BalanceWhiteAuto.getRange();

// Set the current enum value (enable continuous automatic white balance)
cam.BalanceWhiteAuto.set(EnumDefineSet.AutoEntry.CONTINUOUS.getValue());
```

```
// Get the current status of automatic white balance
long value = cam.BalanceWhiteAuto.get();
```

Bool ([BoolFeature](#)) :

Related interfaces:

```
void    set(boolean boolValue);        // Set
boolean get();                          // Get
```

Sample code:

```
// Set the current bool value
cam.LineInverter.set(true);

// Get bool value
boolean lineInverterValue = cam.LineInverter.get();
```

String ([StringFeature](#)) :

Related interfaces:

```
void    set(String value);             // Set
String  get();                         // Get
int      getMaxLength();               // Get the maximum length value of string feature
```

Sample code:

```
// Get the maximum length of string
int maxLength = cam.DeviceUserID.getMaxLength();

//Get the current value of string
String curValue = cam.DeviceUserID.get();

// Set the value of string
cam.DeviceUserID.set("cam0");
```

Buffer([BufferFeature](#)):

Related interfaces:

```
void      set(ByteBuffer data);        // Set
ByteBuffer get();                      // Get
int        getLength();                // Get the length of Buffer feature
```

Sample code:

```
// Get the length of buffer data
int bufferLength = cam.LUTValueAll.getLength();

// Get chunk data(LUT)
ByteBuffer bufferData = cam.LUTValueAll.get();

// Set chunk data(LUT)
```

```
cam.LUTValueAll.set(bufferData);
```

Command ([CommandFeature](#)) :

Related interface:

```
void sendCommand(); // Send command
```

Sample code:

```
// Send command: software trigger  
cam.TriggerSoftware.sendCommand();
```

3.7. Import and export camera configuration

The function to export camera configuration files allows you to save the current values of all camera settable parameters to a local file and import them into the camera when needed. Users can export multiple configuration files and then import different parameters as needed.

Sample code:

```
// Export camera configuration parameter file  
cam.exportConfigFile("config0.txt");  
  
// Import camera configuration parameter file  
cam.importConfigFile("config0.txt", false);
```

3.8. Exceptions

When an exception occurs inside the calling interface function, the error handling mechanism detects and throws different types of exception, and the exception types inherit from **ExceptionSet**.

3.8.1. Sample code

A typical error handling sample code is as follows:

```
Device cam = null;  
Try  
{  
    // When the interface function is called, the function throws an exception  
    // internally  
    Device cam = deviceManager.openDeviceByIndex(1);  
}  
Catch(ExceptionSet exceptionSet)  
{  
    exceptionSet.printStackTrace();  
}
```

The user can also perform classification processing by testing the specific type of error captured:

```
RawImage rawImage = null;  
try{  
    // The parameter is timeout time, and its unit is ms  
    rawImage = dataStream.getRawImage(500);  
}
```

```
catch(ExceptionSet.Timeout timeout) {  
    // Timeout is captured. If the timeout is captured during loop  
    // acquisition, continue directly without printing error messages  
    continue;  
}  
catch(ExceptionSet exceptionSet) {  
    // Other exception is captured. If the exception is captured during loop  
    // acquisition, continue trying to acquire the image or stop acquisition,  
    // after printing the exception messages  
    exceptionSet.printStackTrace();  
    // continue;  
}
```

3.8.2. Exception type

Exception type	Descriptions
UnexpectedError	Unexpected exception
NotFoundTL	Not found TL
NotFoundDevice	Not found device
OffLine	Camera offline
InvalidParameter	Invalid parameter
InvalidHandle	Invalid handle
InvalidCall	Invalid call
InvalidAccess	Invalid access
NeedMoreBuffer	Insufficient buffer
FeatureTypeError	Feature type error
OutOfRange	Out of range
NotImplemented	Function is not implemented
NotInitApi	Not initialized
Timeout	Timeout
ParameterTypeError	Parameter type error
RepeatOpened	Repeat opened
NotSupported	Unsupported function
PixelFormatNotSupport	Unsupported pixel format
OutOfMemory	Out of memory
OnlySupportAndroid	Android support only
SaveImageError	Save image error
SaveImageNoPermission	No permission to save image
BitmapError	Errors in operating bitmap

CpuNotSupportAccelerate	Cpu does not support image processing library acceleration
NativeWindowError	Errors in operating surface

4. FAQ

No.	General Question	Answer
1.	Cannot enumerate U3V devices under Android	Please use the no Root permission enumeration method
2.	How to display the description for each interface in the Android Studio development environment	Import Javadoc, see the section 2.1.3 for details

5. Appendix

5.1. Feature parameter

Note: MER-GEV: Mercury (Gen 1) GEV camera. MER-U3V: Mercury (Gen 1) USB3.0 camera. MER-U2: Mercury (Gen 1) USB2.0 camera. MER2-GEV: Mercury2 (Gen 2) GEV camera.

5.1.1. Device feature parameter

Feature parameter	Explanation	Feature class	Camera model			
DeviceInformation Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV
DeviceVendorName	Device vendor name	StringFeature	√	√	√	√
DeviceModelName	Device model name	StringFeature	√	√	√	√
DeviceFirmwareVersion	Device firmware version	StringFeature	√	√	√	√
DeviceVersion	Device version	StringFeature	√	√	√	√
DeviceSerialNumber	Device serial number	StringFeature	√	√	√	√
FactorySettingVersion	Factory setting version	StringFeature	√	√	√	√
DeviceUserID	User-defined name	StringFeature	√	√	√	√
DeviceLinkSelector	Device link selector, see C Software Development Manual for details	IntFeature	√	√	√	√
DeviceLinkThroughputLimit	Device link throughput limit	IntFeature	√	√	√	√
DeviceLinkThroughputLimitMode	Device link throughput limit mode, see SwitchEntry for details	EnumFeature	√	√	√	√
DeviceLinkCurrentThroughput	Current device link throughput	IntFeature	√	√	√	√
DeviceReset	Device reset	CommandFeature				√
TimestampTickFrequency	Timestamp tick frequency	IntFeature				√
TimestampLatch	Timestamp latch	CommandFeature				√
TimestampReset	Timestamp reset	CommandFeature				√
TimestampLatchReset	Timestamp latch reset	CommandFeature				√
TimestampLatchValue	Timestamp latch value	IntFeature				√
ImageFormat Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV
SensorWidth	Sensor width	IntFeature	√	√	√	√
SensorHeight	Sensor height	IntFeature	√	√	√	√
WidthMax	Maximum width	IntFeature	√	√	√	√
HeightMax	Maximum height	IntFeature	√	√	√	√

OffsetX	Horizontal offset	IntFeature	√	√	√	√
OffsetY	Vertical offset	IntFeature	√	√	√	√
Width	Image width	IntFeature	√	√	√	√
Height	Image height	IntFeature	√	√	√	√
BinningHorizontal	Horizontal Binning	IntFeature	√	√	√	√
BinningVertical	Vertical Binning	IntFeature	√	√	√	√
DecimationHorizontal	Horizontal decimation	IntFeature	√	√	√	√
DecimationVertical	Vertical decimation	IntFeature	√	√	√	√
PixelSize	Pixel size, see PixelSizeEntry for details	EnumFeature	√	√	√	√
PixelColorFilter	Bayer format, see PixelColorFilterEntry for details	EnumFeature	√	√	√	√
PixelFormat	Pixel format, see PixelFormatEntry for details	EnumFeature	√	√	√	√
ReverseX	Horizontal reverse	BoolFeature	√	√	√	√
ReverseY	Vertical reverse	BoolFeature	√	√	√	√
TestPattern	Test pattern, see TestPatternEntry for details	EnumFeature	√	√	√	√
TestPatternGeneratorSelector	Test pattern generator selector, see C Software Development Manual and TestPatternGeneratorSelectorEntry for details	EnumFeature	√	√	√	√
RegionSendMode	ROI send mode, see RegionSendModeEntry for details	EnumFeature	√	√	√	√
RegionMode	Region mode, see SwitchEntry for details	EnumFeature	√	√	√	√
RegionSelector	Region selector, see C Software Development Manual and RegionSelectorEntry for details	EnumFeature	√	√	√	√
CenterWidth	Center width	IntFeature		√ *		
CenterHeight	Center height	IntFeature		√ *		
BinningHorizontalMode	Horizontal Binning mode, see BinningHorizontalModeEntry for details	EnumFeature				√
BinningVerticalMode	Vertical Binning mode, see BinningVerticalModeEntry for details	EnumFeature				√
TransportLayer Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV
PayloadSize	Payload size	IntFeature	√	√	√	√
CenterWidth	Center width	IntFeature	√	√	√	√
CenterHeight	Center height	IntFeature	√	√	√	√

GevCurrentIPConfiguratio nLLA	Configuring IP in LLA mode	BoolFeature	√			√
GevCurrentIPConfiguratio nDHCP	Configuring IP in DHCP mode	BoolFeature	√			√
GevCurrentIPConfiguratio nPersistentIP	Configuring IP in permanent IP mode	BoolFeature	√			√
EstimatedBandwidth	Estimated bandwidth	IntFeature	√			√
GevHeartbeatTimeout	Heartbeat timeout time	IntFeature	√			√
GevSCPSPacketSize	Stream channel packet size	IntFeature	√			√
GevSCPD	Stream channel packet delay	IntFeature	√			√
GevLinkSpeed	Link speed	IntFeature	√			√
DigitalIO Section			MER- GEV	MER- U3V	MER- U2	MER2 -GEV
UserOutputSelector	User-defined output selector, see C Software Development Manual and UserOutputSelectorEntry for details	EnumFeature	√	√	√	√
UserOutputValue	User-defined output value	BoolFeature	√	√	√	√
UserOutputMode	User IO output mode, see UserOutputModeEntry for details	EnumFeature			√	
StrobeSwitch	Strobe switch, see SwitchEntry for details	EnumFeature			√	
LineSelector	Line selector, see C Software Development Manual and LineSelectorEntry for details	EnumFeature	√	√	√	√
LineMode	Line mode, see LineModeEntry for details	EnumFeature	√	√	√	√
LineSource	Line output source, see LineSourceEntry for details	EnumFeature	√	√	√	√
LineInverter	Line level inversion	BoolFeature	√	√	√	√
LineStatus	Line status	BoolFeature	√	√	√	√
LineStatusAll	Status of all lines	IntFeature	√	√	√	√
AnalogControls Section			MER- GEV	MER- U3V	MER- U2	MER2 -GEV
GainAuto	Automatic gain, see AutoEntry for details	EnumFeature	√	√	√	√
GainSelector	Gain channel selector, see C Software Development Manual and GainSelectorEntry for details	EnumFeature	√	√	√	√
BlackLevelAuto	Automatic black level, see AutoEntry for details	EnumFeature	√	√	√	√
BlackLevelSelector	Black level channel selector, see C Software Development Manual and BlackLevelSelectEntry for details	EnumFeature	√	√	√	√
BalanceWhiteAuto	Automatic white balance, see AutoEntry for details	EnumFeature	√	√	√	√
BalanceRatioSelector	White balance channel	EnumFeature	√	√	√	√

	selector, see C Software Development Manual and BalanceRatioSelectorEntry for details					
BalanceRatio	White balance ratio	FloatFeature	√	√	√	√
DeadPixelCorrect	Dead pixel correction	EnumFeature	√	√	√	√
Gain	Gain	FloatFeature	√	√	√	√
BlackLevel	Black level	FloatFeature	√	√	√	√
GammaEnable	Gamma enable	BoolFeature				√
GammaMode	Gamma mode, see GammaModeEntry for details	EnumFeature				√
Gamma	Gamma	FloatFeature				√
DigitalShift	Digital shift	IntFeature				√
CustomFeature Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV
ADCLLevel	AD conversion level	IntFeature			√	
HBlanking	Horizontal blanking	IntFeature			√	
VBlanking	Vertical blanking	IntFeature			√	
UserPassword	User encryption zone password	StringFeature			√	
VerifyPassword	User encryption zone verify password	StringFeature			√	
UserData	User encryption zone content	BufferFeature			√	
ExpectedGrayValue	Expected gray value	IntFeature	√	√	√	√
AALightEnvironment	Auto exposure, auto gain, lighting environment type, see AALightEnvironmentEntry for details	EnumFeature			√	
ImageGrayRaiseSwitch	Image gray value raise switch, see SwitchEntry for details	EnumFeature			√	
AAROIOffsetX	Auto adjust the X offset of ROI	IntFeature	√	√	√	√
AAROIOffsetY	Auto adjust the Y offset of ROI	IntFeature	√	√	√	√
AAROIWidth	Auto adjust the width of ROI	IntFeature	√	√	√	√
AAROIHeight	Auto adjust the height of ROI	IntFeature	√	√	√	√
AutoGainMin	Minimum automatic gain	FloatFeature	√	√	√	√
AutoGainMax	Maximum automatic gain	FloatFeature	√	√	√	√
AutoExposureTimeMin	Minimum automatic exposure	FloatFeature	√	√	√	√
AutoExposureTimeMax	Maximum automatic exposure	FloatFeature	√	√	√	√
ContrastParam	Contrast parameter	IntFeature	√	√	√	√
ColorCorrectionParam	Color correction parameter	IntFeature	√	√	√	√
AWBROIOffsetX	The X coordinate of automatic white balance ROI	IntFeature	√	√	√	√
AWBROIOffsetY	The Y coordinate of	IntFeature	√	√	√	√

	automatic white balance ROI					
AWBROIWidth	The width of automatic white balance ROI	IntFeature	√	√	√	√
AWBROIHeight	The height of automatic white balance ROI	IntFeature	√	√	√	√
GammaParam	Gamma parameter	FloatFeature	√	√	√	√
AWBLampHouse	Automatic white balance lamp house, see AWBLampHouseEntry for details	EnumFeature	√	√	√	√
SharpnessMode	Sharpening mode, see SwitchEntry for details	EnumFeature	√	√	√	√
Sharpness	Sharpness	FloatFeature	√	√	√	√
FrameInformation	Image frame information	BufferFeature			√	
UserSetControl Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV
UserSetLoad	Load the user set	CommandFeature	√	√	√	√
UserSetSave	Save the user set	CommandFeature	√	√	√	√
UserSetSelector	User set selector, see C Software Development Manual and UserSetEntry for details	EnumFeature	√	√	√	√
UserSetDefault	Default the user set, see UserSetEntry for details	EnumFeature	√	√	√	√
LUT Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV
LUTValueAll	LUT content	BufferFeature	√	√	√	√
LUTSelector	LUT selector, see C Software Development Manual and LutSelectorEntry for details	EnumFeature	√	√	√	√
LUTEnable	LUT enable	BoolFeature				√
LUTIndex	LUT index	IntFeature				√
LUTValue	LUT value	IntFeature				√
Color Transformation Control			MER-GEV	MER-U3V	MER-U2	MER2-GEV
ColorTransformationMode	Color transformation mode, see ColorTransformationModeEntry for details	EnumFeature				√
ColorTransformationEnable	Color transformation enable	BoolFeature				√
ColorTransformationValueSelector	Color transformation matrix value selector, see ColorTransformationValueSelectorEntry for details	EnumFeature				√
ColorTransformationValue	Color transformation matrix value	FloatFeature				√

ChunkData Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV
ChunkModeActive	Chunk data enable	BoolFeature	√	√	√	√
ChunkEnable	Single chunk data enable	BoolFeature	√	√	√	√
ChunkSelector	Chunk data selector, see C Software Development Manual and ChunkSelectorEntry for details	EnumFeature	√	√	√	√
Device Feature			MER-GEV	MER-U3V	MER-U2	MER2-GEV
DeviceCommandTimeout	Device command timeout	IntFeature	√			√
DeviceCommandRetryCount	Device command retry count	IntFeature	√			√
AcquisitionControl Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV
FrameBufferOverwriteActive	Frame buffer overwrite enable	BoolFeature	√	√	√	√
TriggerSoftware	Software trigger	CommandFeature	√	√	√	√
TransferStart	Start transfer	CommandFeature	√	√	√	√
AcquisitionMode	Acquisition mode, see AcquisitionModeEntry for details	EnumFeature	√	√	√	√
TriggerMode	Trigger mode, see SwitchEntry for details	EnumFeature	√	√	√	√
TriggerActivation	Trigger activation, see TriggerActivationEntry for details	EnumFeature	√	√	√	√
ExposureAuto	Auto exposure, see AutoEntry for details	EnumFeature	√	√	√	√
TriggerSource	Trigger source, see TriggerSourceEntry for details	EnumFeature	√	√	√	√
ExposureMode	Exposure mode, see ExposureModeEntry for details	EnumFeature	√	√	√	√
TriggerSelector	Trigger type selector, see C Software Development Manual and TriggerSelectorEntry for details	EnumFeature	√	√	√	√
TransferControlMode	Transfer control mode, see TransferControlModeEntry for details	EnumFeature	√	√	√	√
TransferOperationMode	Transfer operation mode, see TransferOperationModeEntry for details	EnumFeature	√	√	√	√
AcquisitionFrameRateMode	Acquisition frame rate adjustment mode, see SwitchEntry for details	EnumFeature	√	√	√	√
FixedPatternNoiseCorrectMode	Fixed pattern noise correct mode, see SwitchEntry for details	EnumFeature	√	√	√	√
ExposureTime	ExposureTime	FloatFeature	√	√	√	√
TriggerFilterRaisingEdge	Raising edge trigger filter	FloatFeature	√	√	√	√
TriggerFilterFallingEdge	Falling edge trigger filter	FloatFeature	√	√	√	√

TriggerDelay	Trigger delay	FloatFeature	√	√	√	√
AcquisitionFrameRate	Acquisition frame rate	FloatFeature	√	√	√	√
CurrentAcquisitionFrameRate	Current acquisition frame rate	FloatFeature	√	√	√	√
TransferBlockCount	Transfer block count	IntFeature	√	√	√	√
TriggerSwitch	External trigger switch, see SwitchEntry for details	EnumFeature			√	
AcquisitionSpeedLevel	Acquisition speed level	IntFeature			√	
AcquisitionFrameCount	Acquisition frame count	IntFeature			√	
AcquisitionBurstFrameCount	Acquisition burst frame count	IntFeature				√
AcquisitionStatusSelector	Acquisition status selector, see C Software Development Manual and AcquisitionStatusSelectorEntry for details	EnumFeature				√
AcquisitionStatus	Acquisition status	BoolFeature				√
ExposureDelay	Exposure delay	FloatFeature				√
CounterAndTimerControl Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV
TimerSelector	Selects which Counter to configure , see TimerSelectorEntry for details	EnumFeature		√		
TimerDuration	Sets the duration (in microseconds) of the Timer pulse	FloatFeature		√		
TimerDelay	Sets the duration (in microseconds) of the delay to apply at the reception of a trigger before starting the Timer	FloatFeature		√		
TimerTriggerSource	Selects the source of the trigger to start the Timer , see TimerTriggerSourceEntry for details	EnumFeature		√		
CounterSelector	Selects which Counter to configure , see CounterSelectorEntry for details	EnumFeature		√		
CounterEventSource	Selects the events that will be the source to increment the Counter , see CounterEventSourceEntry for details	EnumFeature		√		
CounterResetSource	Selects the signals that will be the source to reset the Counter , see CounterResetSourceEntry for details	EnumFeature		√		
CounterResetActivation	Selects the Activation mode of the Counter Reset Source signal ,	EnumFeature		√		

	see CounterResetActivationEntry for details				
CounterReset	Does a software reset of the selected Counter and starts it	CommandFeature		√	

CenterWidth, CenterHeight (√ *) only support binocular cameras based on MER-U3V.

5.1.2. Stream feature parameter

Feature parameter	Explanation	Feature class	MER-GEV	MER-U3V	MER-U2	MER2-GEV
StreamAnnouncedBufferCount	Announced Buffer count	IntFeature	√	√	√	√
StreamDeliveredFrameCount	Delivered frame count (including incomplete frames)	IntFeature	√	√	√	√
StreamLostFrameCount	The number of lost frames caused by insufficient Buffer	IntFeature	√	√	√	√
StreamIncompleteFrameCount	Delivered incomplete frame count	IntFeature	√	√	√	√
StreamDeliveredPacketCount	Delivered packet count	IntFeature	√	√	√	√
StreamResendPacketCount	Resend packet count	IntFeature	√			√
StreamRescuedPacketCount	Rescued packet count	IntFeature	√			√
StreamResendCommandCount	Resend command count	IntFeature	√			√
StreamUnexpectedPacketCount	Unexpected packet count	IntFeature	√			√
MaxPacketCountInOneBlock	Maximum resend packet count in one block	IntFeature	√			√
MaxPacketCountInOneCommand	Maximum packet count in one resend command	IntFeature	√			√
ResendTimeout	Resend timeout time	IntFeature	√			√
MaxWaitPacketCount	Maximum waiting packet count	IntFeature	√			√
ResendMode	Resend mode, see SwitchEntry for details	EnumFeature	√			√
StreamMissingBlockIDCount	Missing BlockID count	IntFeature	√			√
BlockTimeout	Data block timeout time	IntFeature	√			√
MaxNumQueueBuffer	Maximum number of Buffer in the acquisition queue	IntFeature	√			√
PacketTimeout	Packet timeout time	IntFeature	√			√
StreamTransferSize	Transfer data block size	IntFeature		√		
StreamTransferNumberUrb	Transfer data block number	IntFeature		√		

5.2. Function class definition

5.2.1. Feature

It is responsible for checking the basic functions of various data type functions: get the function name, test if they are implemented, readable and writable.

The Feature class is the parent of the [IntFeature](#)/[FloatFeature](#)/[EnumFeature](#)/[BoolFeature](#)/[StringFeature](#)/[BufferFeature](#)/ [CommandFeature](#).

◆ Interface list

String	getName();	The string of getting function name
boolean	isImplemented();	Test if the feature is implemented
boolean	isReadable();	Test if the feature is readable
boolean	isWritable ();	Test if the feature is writable

Sample code:

```
// Get function name
String strName = cam.StreamTransferSize.getName();

// Whether the function is implemented
boolean isImplemented = cam.StreamTransferSize.isImplemented();
if(isImplemented)
{
    // Is this writable
    boolean isWritable = cam.StreamTransferSize.isWritable();
    if(isWritable)
    {
        // Set the size of URB
        cam.StreamTransferSize.set(512*1024);
    }

    // Is this readable
    boolean isReadable = cam.StreamTransferSize.isReadable();
    if(isReadable)
    {
        // Get the current size of URB
        long transferSize = cam.StreamTransferSize.get();
    }
}
```

◆ Interface description

➤ getName

Declarations:

String Feature.getName();

Descriptions:

Get function name

Returns:

true: Implemented
false: Unimplemented

Exceptions:

Getting function name fails, return "".

➤ isImplemented**Declarations:**

```
boolean Feature.isImplemented();
```

Descriptions:

Test if the feature parameter is implemented

Returns:

true: Implemented
false: Unimplemented

Exceptions:

- 1) If the feature parameter is invalid, return false.
- 2) If getting whether the feature parameter is implemented fails due to other reasons, an exception is thrown. See [section 3.8.2](#) for details.

➤ isReadable**Declarations:**

```
boolean Feature.isReadable();
```

Descriptions:

Test if the feature parameter is readable

Returns:

true: Readable
false: Unreadable

Exceptions:

- 1) If the function is not implemented, return false.
- 2) If getting whether the feature parameter is readable fails, an exception is thrown. See [section 3.8.2](#) for details.

➤ **isWritable**

Declarations:

```
boolean Feature.isWritable();
```

Descriptions:

Test if the feature parameter is writable

Returns:

true: Writable
False: Unwritable

Exceptions:

- 1) If the function is not implemented, return false.
- 2) If getting whether the feature parameter is writable fails, an exception is thrown. See [section 3.8.2](#) for details.

5.2.2. IntFeature

It is responsible for checking and controlling the camera's int feature, inherited from the [Feature](#) class.

◆ **Interface list**

String	getName();	The string of getting function name
boolean	isImplemented();	Test if the feature is implemented
boolean	isReadable();	Test if the feature is readable
boolean	isWritable ();	Test if the feature is writable
long	getMin();	Get the minimum parameter
long	getMax();	Get the maximum parameter
long	getInc();	Get the step of the parameter
long	get();	Get the value of the parameter
void	set(long value);	Set the value of the parameter

Sample code:

```
// Get the maximum image width  
long max = cam.Width.getMax();
```

```
// Set the current image width to the maximum  
cam.Width.set(max);  
  
// Get the width of the current image  
long value = cam.Width.get();
```

◆ Interface description

➤ getMin

Declarations:

```
long IntFeature.getMin();
```

Descriptions:

Get the minimum

Returns:

Minimum

Exceptions:

If getting the minimum fails, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ getMax

Declarations:

```
long IntFeature.getMax();
```

Descriptions:

Get the maximum

Returns:

Maximum

Exceptions:

If getting the maximum fails, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ getStep

Declarations:

```
long IntFeature.getStep();
```

Descriptions:

Get the value of step

Returns:

Step

Exceptions:

If getting the value of step fails, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ get**Declarations:**

```
IntFeature.get();
```

Descriptions:

Get the int value

Returns:

The int value got

Exceptions:

If getting the int value unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ set**Declarations:**

```
void IntFeature.set(long value);
```

Parameters:

[in] value **Set the int value**

Returns:

Int value

Exceptions:

If setting the int feature unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

5.2.3. FloatFeature

It is responsible for checking and controlling the camera's float feature, inherited from the [Feature](#) class.

◆ Interface list

String	getName();	The string of getting function name
boolean	isImplemented();	Test if the feature is implemented
boolean	isReadable();	Test if the feature is readable
boolean	isWritable ();	Test if the feature is writable
double	getMin();	Get the minimum value of the parameter
double	getMax();	Get the maximum value of the parameter
boolean	hasInc();	Test if there is a step
double	getInc();	Get the step of the parameter
String	getUnit();	Get the unit
double	get();	Get the parameter value
void	set(double value);	Set the parameter value

Sample code:

```
// Get the maximum settable value of the exposure time
double max = cam.ExposureTime.getMax();

// Set the current exposure time to 10ms
cam.ExposureTime.set(10000);

// Get the current value of exposure time
double curValue = cam.ExposureTime.get();
```

◆ Interface description

➤ getMin

Declarations:

```
double FloatFeature.getMin();
```

Descriptions:

Get the minimum

Returns:

Minimum

Exceptions:

If getting the minimum fails, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **getMax**

Declarations:

```
double FloatFeature.getMax();
```

Descriptions:

Get the maximum

Returns:

Maximum

Exceptions:

If getting the maximum fails, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **hasInc**

Declarations:

```
boolean FloatFeature.hasInc();
```

Descriptions:

Test if the step is supported

Returns:

If the step is supported

Exceptions:

If the execution fails, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **getInc**

Declarations:

```
double FloatFeature.getInc();
```

Descriptions:

Get the step

Returns:

Step

Exceptions:

If getting the value of step fails, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **getUnit**

Declarations:

```
String FloatFeature.getUnit();
```

Descriptions:

Get the unit

Returns:

Unit

Exceptions:

If getting the unit fails, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **get**

Declarations:

```
double FloatFeature.get();
```

Descriptions:

Get the value of float feature

Returns:

The float feature value got

Exceptions:

If getting the float feature value unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **set**

Declarations:

```
void FloatFeature.set(double value);
```

Descriptions:

Set the value of float feature

Parameters:

[in] value The set value of float

Exceptions:

If setting the float feature unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

5.2.4. EnumFeature

It is responsible for checking and controlling the camera's enum feature, inherited from the [Feature](#) class.

◆ Interface list

String	getName();	The string of getting function name
boolean	isImplemented();	Test if the feature is implemented
boolean	isReadable();	Test if the feature is readable
boolean	isWritable ();	Test if the feature is writable
List<EnumDescription>	getRange();	Get the list of supported enum
long	get();	Get the parameter value
void	set(long value);	Set the parameter value

The information in EnumDescription are as follows:

value	Enum value
symbolic	The description of enum

Sample code:

```
// Get the settable range of the enum value
List<EnumDescription> enumDescList = cam.BalanceWhiteAuto.getRange();

// Set the current enum value (enable continuous automatic white balance)
cam.BalanceWhiteAuto.set(CONTINUOUS.getValue());

// Get the status of the current automatic white balance
long value = cam.BalanceWhiteAuto.get();
```

◆ Interface description

➤ **getRange**

Declarations:

```
List<EnumDescription> EnumFeature.getRange();
```

Descriptions:

Get the list of supported enum

Returns:

The list of supported enum

Exceptions:

If getting the range of enmu feature unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **get**

Declarations:

```
long EnumFeature.get();
```

Descriptions:

Get the value of the enum feature

Returns:

The value of the enum feature

Exceptions:

If getting enmu feature unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **set**

Declarations:

```
void EnumFeature.set( long enumValue);
```

Descriptions:

Get the value of the enum feature

Parameters:

[in] enumValue The set value of enmu

Exceptions:

If setting enmu feature unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

5.2.5. BoolFeature

It is responsible for checking and controlling the camera's bool feature, inherited from the [Feature](#) class.

◆ Interface list

String	getName();	The string of getting function name
boolean	isImplemented();	Test if the feature is implemented
boolean	isReadable();	Test if the feature is readable
boolean	isWritable ();	Test if the feature is writable
boolean	get();	Get the parameter value
void	set(boolean value);	Set the parameter value

Sample code:

```
// Set the current value of bool
cam.LineInverter.set(true);

// Get the value of bool
boolean value = cam.LineInverter.get();
```

◆ Interface description**➤ get****Declarations:**

boolean BoolFeature.get();

Descriptions:

Get the value of bool feature

Returns:

The bool value got

Exceptions:

If getting bool value unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ set

Declarations:

```
void BoolFeature.set(boolean value);
```

Descriptions:

Set the value of bool feature

Parameters:

[in] value The value of bool

Exceptions:

If setting bool value unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

5.2.6. StringFeature

It is responsible for checking and controlling the camera's string feature, inherited from the [Feature](#) class.

◆ Interface list

String	getName();	The string of getting function name
boolean	isImplemented();	Test if the feature is implemented
boolean	isReadable();	Test if the feature is readable
boolean	isWritable ();	Test if the feature is writable
int	getMaxLength();	Get the maximum length that a string can be set
String	get();	Get the parameter value
void	set(String value);	Set the parameter value

Sample code:

```
// Get the maximum length of string feature
int maxLength = cam.DeviceUserID.getMaxLength();

// Get the current value of string feature
String curValue = cam.DeviceUserID.get();

// Set the value of string feature
cam.DeviceUserID.set("MyUserID");
```

◆ Interface description

➤ getMaxLength

Declarations:

```
int StringFeature.getMaxLength();
```

Descriptions:

Get the maximum length that a string feature can be set

Returns:

The maximum length that a string feature can be set

Exceptions:

If getting the maximum length of the string feature unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **get**

Declarations:

```
String StringFeature.get();
```

Descriptions:

Get the value of string

Returns:

The string value got

Exceptions:

If getting the string value unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **set**

Declarations:

```
void StringFeature.set(String value);
```

Descriptions:

Get the value of string

Parameters:

[in] value The set value of string

Exceptions:

If setting the string value unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

5.2.7. BufferFeature

It is responsible for checking and controlling the camera's Buffer, inherited from the [Feature](#) class.

◆ Interface list

String	getName();	The string of getting function name
boolean	isImplemented();	Test if the feature is implemented
boolean	isReadable();	Test if the feature is readable
boolean	isWritable ();	Test if the feature is writable
int	getLength();	Get the length of the Buffer
ByteBuffer	get();	Get the parameter value
void	set(ByteBuffer value);	Set the parameter value

Sample code:

```
// Get the length of buffer data
int bufferLength = cam.LUTValueAll.getLength();

// Get the chunk data(LUT)
ByteBuffer bufferData = cam.LUTValueAll.get();

// Set the chunk data(LUT)
cam.LUTValueAll.set(bufferData);
```

◆ Interface description**➤ getLength****Declarations:**

```
int BufferFeature.getLength();
```

Descriptions:

Get the length of the Buffer

Returns:

The length of the Buffer

Exceptions:

If getting the length of the Buffer unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ get

Declarations:

```
ByteBuffer BufferFeature.get();
```

Descriptions:

Get the data of Buffer

Returns:

ByteBuffer object

Exceptions:

If getting the data of Buffer unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ set

Declarations:

```
void BufferFeature.set(ByteBuffer value);
```

Descriptions:

Set the data of Buffer

Parameters:

[in] value **Set the value of the Buffer**

Exceptions:

If setting the Buffer unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

5.2.8. CommandFeature

It is responsible for the command feature function of the camera, inherited from the [Feature](#) class.

◆ Interface list

String	getName();	The string of getting function name
boolean	isImplemented();	Test if the feature is implemented
boolean	isReadable();	Test if the feature is readable
boolean	isWritable ();	Test if the feature is writable

void	sendCommand();	Send command
------	----------------	--------------

Sample code:

```
// Send software trigger command
cam.TriggerSoftware.sendCommand();
```

◆ Interface description

➤ sendCommand

Declarations:

```
void CommandFeature.sendCommand();
```

Descriptions:

Send command

Exceptions:

If sending the command unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

5.3. Data type definition

5.3.1. DeviceClass

Definition	Value	Explanation
UNKNOWN	0	Unknown device type
USB2	1	USB2.0 camera
GEV	2	GEV camera (GigE Vision)
U3V	3	USB3.0 camera (USB3 Vision)

5.3.2. AccessStatus

Definition	Value	Explanation
UNKNOWN	0	The current status of the device is unknown
READWRITE	1	The device is currently readable and writable
READONLY	2	The device currently only supports reading
NOACCESS	3	The device currently does not support reading and writing

5.3.3. AccessMode

Definition	Value	Explanation
------------	-------	-------------

READONLY	2	Open the device in read-only mode
CONTROL	3	Open the device in control mode
EXCLUSIVE	4	Open the device in exclusive mode

5.3.4. FrameStatus

Definition	Value	Explanation
SUCCESS	0	Normal frame
IMCOMPLETE	-1	Incomplete frame
INVALID_IMAGE_INFO	-2	Image header information is invalid

5.3.5. PixelFormatEntry

Definition	Value	Explanation
UNDEFINED	0x00000000	Undefined
MONO8	0x01080001	Monochrome 8-bit
MONO8_SIGNED	0x01080002	Monochrome 8-bit signed
MONO10	0x01100003	Monochrome 10-bit unpacked
MONO12	0x01100005	Monochrome 12-bit unpacked
MONO14	0x01100025	Monochrome 14-bit unpacked
MONO16	0x01100007	Monochrome 16-bit
BAYER_GR8	0x01080008	Bayer Green-Red 8-bit
BAYER_RG8	0x01080009	Bayer Red-Green 8-bit
BAYER_GB8	0x0108000A	Bayer Green-Blue 8-bit
BAYER_BG8	0x0108000B	Bayer Blue-Green 8-bit
BAYER_GR10	0x0110000C	Bayer Green-Red 10-bit
BAYER_RG10	0x0110000D	Bayer Red-Green 10-bit
BAYER_GB10	0x0110000E	Bayer Green-Blue 10-bit
BAYER_BG10	0x0110000F	Bayer Blue-Green 10-bit
BAYER_GR12	0x01100010	Bayer Green-Red 12-bit
BAYER_RG12	0x01100011	Bayer Red-Green 12-bit
BAYER_GB12	0x01100012	Bayer Green-Blue 12-bit
BAYER_BG12	0x01100013	Bayer Blue-Green 12-bit
BAYER_GR16	0x0110002E	Bayer Green-Red 16-bit
BAYER_RG16	0x0110002F	Bayer Red-Green 16-bit
BAYER_GB16	0x01100030	Bayer Green-Blue 16-bit

BAYER_BG16	0x01100031	Bayer Blue-Green 16-bit
RGB8_PLANAR	0x02180021	Red-Green-Blue 8-bit planar
RGB10_PLANAR	0x02300022	Red-Green-Blue 10-bit planar
RGB12_PLANAR	0x02300023	Red-Green-Blue 12-bit planar
RGB16_PLANAR	0x02300024	Red-Green-Blue 16-bit planar

5.3.6. PixelSizeEntry

Definition	Value	Explanation
BPP8	8	Pixel size of BPP8
BPP10	10	Pixel size of BPP10
BPP12	12	Pixel size of BPP12
BPP16	16	Pixel size of BPP16
BPP24	24	Pixel size of BPP24
BPP30	30	Pixel size of BPP30
BPP32	32	Pixel size of BPP32
BPP36	36	Pixel size of BPP36
BPP48	48	Pixel size of BPP48
BPP64	64	Pixel size of BPP64

5.3.7. PixelColorFilterEntry

Definition	Value	Explanation
NONE	0	None
BAYER_RG	1	RG format
BAYER_GB	2	GB format
BAYER_GR	3	GR format
BAYER_BG	4	BG format

5.3.8. AcquisitionModeEntry

Definition	Value	Explanation
SINGLE_FRAME	0	Single frame mode
MULTI_FRAME	1	Multi-frame mode
CONTINUOUS	2	Continuous mode

5.3.9. TriggerSourceEntry

Definition	Value	Explanation
------------	-------	-------------

SOFTWARE	0	Software trigger
LINE0	1	Trigger source 0
LINE1	2	Trigger source 1
LINE2	3	Trigger source 2
LINE3	4	Trigger source 3

5.3.10. TriggerActivationEntry

Definition	Value	Explanation
FALLING_EDGE	0	Falling edge trigger
RISING_EDGE	1	Rising edge trigger

5.3.11. ExposureModeEntry

Definition	Value	Explanation
TIMED	1	Exposure time register controls exposure time
TRIGGER_WIDTH	2	Trigger signal width controls exposure time

5.3.12. UserOutputSelectorEntry

Definition	Value	Explanation
OUTPUT0	1	Output 0
OUTPUT1	2	Output 1
OUTPUT2	4	Output 2

5.3.13. UserOutputModeEntry

Definition	Value	Explanation
STROBE	0	Strobe
USER_DEFINED	1	User defined

5.3.14. GainSelectorEntry

Definition	Value	Explanation
ALL	0	All gain channels
RED	1	Red channel gain
GREEN	2	Green channel gain
BLUE	3	Blue channel gain

5.3.15. BlackLevelSelectEntry

Definition	Value	Explanation
------------	-------	-------------

ALL	0	All black level channels
RED	1	Red channel black level
GREEN	2	Green channel black level
BLUE	3	Blue channel black level

5.3.16. BalanceRatioSelectorEntry

Definition	Value	Explanation
RED	0	Red channel
GREEN	1	Green channel
BLUE	2	Blue channel

5.3.17. AALightEnvironmentEntry

Definition	Value	Explanation
NATURE_LIGHT	0	Natural light
AC50HZ	1	50 Hz fluorescent lamp
AC60HZ	2	60 Hz fluorescent lamp

5.3.18. UserSetEntry

Definition	Value	Explanation
DEFAULT	0	Default parameter set
USER_SET0	1	User parameter set 0

5.3.19. AWBLampHouseEntry

Definition	Value	Explanation
ADAPTIVE	0	Adaptive light source
D65	1	The designated color temperature is 6500k
FLUORESCENCE	2	Designated fluorescent lamp
INCANDESCENT	3	Designated incandescent lamp
D75	4	The designated color temperature is 7500k
D50	5	The designated color temperature is 5000k
U30	6	The designated color temperature is 3000k

5.3.20. TestPatternEntry

Definition	Value	Explanation
OFF	0	Off
GRAY_FRAME_RAMP_MOVING	1	Still grayscale increment

SLANT_LINE_MOVING	2	Rolling diagonal stripes
VERTICAL_LINE_MOVING	3	Rolling vertical stripes
HORIZONTAL_LINE_MOVING	4	Rolling horizontal stripes
GREY_VERTICAL_RAMP	5	Grey vertical stripes
SLANT_LINE	6	Static slant stripes

5.3.21. TriggerSelectorEntry

Definition	Value	Explanation
FRAME_START	1	Get one frame
FRAME_BURST_START	2	Start the frame burst acquisition

5.3.22. LineSelectorEntry

Definition	Value	Explanation
LINE0	0	Pin 0
LINE1	1	Pin 1
LINE2	2	Pin 2
LINE3	3	Pin 3
LINE4	4	Pin 4
LINE5	5	Pin 5

5.3.23. LineModeEntry

Definition	Value	Explanation
INPUT	0	Input
OUTPUT	1	Output

5.3.24. LineSourceEntry

Definition	Value	Explanation
OFF	0	Off
STROBE	1	Strobe
USER_OUTPUT0	2	User-defined output 0
USER_OUTPUT1	3	User-defined output 1
USER_OUTPUT2	4	User-defined output 2
EXPOSURE_ACTIVE	5	Active exposure
FRAME_TRIGGER_WAIT	6	Single frame trigger waiting
ACQUISITION_TRIGGER_WAIT	7	Multi-frame trigger waiting

5.3.25. LutSelectorEntry

Definition	Value	Explanation
LUMINANCE	0	Luminance

5.3.26. TransferControlModeEntry

Definition	Value	Explanation
BASIC	0	Basic mode
USER_CONTROLLED	1	User control mode

5.3.27. TransferOperationModeEntry

Definition	Value	Explanation
MULTI_BLOCK	0	Designated the number of frames to send

5.3.28. TestPatternGeneratorSelectorEntry

Definition	Value	Explanation
SENSOR	0	The test image of sensor
REGION0	1	The test image of FPGA

5.3.29. ChunkSelectorEntry

Definition	Value	Explanation
FRAME_ID	1	Frame ID
TIME_STAMP	2	Timestamp

5.3.30. BinningHorizontalModeEntry

Definition	Value	Explanation
SUM	0	The response from combine horizontal photo-sensitive cells will be added
AVERAGE	1	The response from combine horizontal photo-sensitive cells will be averaged

5.3.31. BinningVerticalModeEntry

Definition	Value	Explanation
SUM	0	The response from combine vertical photo-sensitive cells will be added
AVERAGE	1	The response from combine vertical photo-sensitive cells will be averaged

5.3.32. AcquisitionStatusSelectorEntry

Definition	Value	Explanation
ACQUISITION_TRIGGER_WAIT	0	Acquisition trigger waiting

FRAME_TRIGGER_WAIT	1	Frame trigger waiting
--------------------	---	-----------------------

5.3.33. GammaModeEntry

Definition	Value	Explanation
SRGB	0	Default Gamma correction
USER	1	User-defined Gamma correction

5.3.34. ColorTransformationModeEntry

Definition	Value	Explanation
RGB_TO_RGB	0	Default color correction
USER	1	User-defined color correction

5.3.35. ColorTransformationValueSelectorEntry

Definition	Value	Explanation
GAIN00	0	The gain value of color transformation component GAIN00
GAIN01	1	The gain value of color transformation component GAIN01
GAIN02	2	The gain value of color transformation component GAIN02
GAIN10	3	The gain value of color transformation component GAIN10
GAIN11	4	The gain value of color transformation component GAIN11
GAIN12	5	The gain value of color transformation component GAIN12
GAIN20	6	The gain value of color transformation component GAIN20
GAIN21	7	The gain value of color transformation component GAIN21
GAIN22	8	The gain value of color transformation component GAIN22

5.3.36. AutoEntry

Definition	Value	Explanation
OFF	0	Off
CONTINUOUS	1	Continuous
ONCE	2	Once

5.3.37. SwitchEntry

Definition	Value	Explanation
OFF	0	Off
ON	1	On

5.3.38. RegionSendModeEntry

Definition	Value	Explanation
SINGLE_ROI	0	Single ROI
MULTI_ROI	1	Multiple ROI

5.3.39. RegionSelectorEntry

Definition	Value	Explanation
REGION0	0	Region 0
REGION1	1	Region 1
REGION2	2	Region 2
REGION3	3	Region 3
REGION4	4	Region 4
REGION5	5	Region 5
REGION6	6	Region 6
REGION7	7	Region 7

5.3.40. BayerConvertType

Definition	Value	Explanation
NEIGHBOUR	0	Neighborhood average interpolation algorithm
ADAPTIVE	1	Edge adaptive interpolation algorithm
NEIGHBOUR3	2	Neighborhood average interpolation algorithm for larger regions

5.3.41. ValidBit

Definition	Value	Explanation
BIT0_7	0	0-7 bits
BIT1_8	1	1-8 bits
BIT2_9	2	2-9 bits
BIT3_10	3	3-10 bits
BIT4_11	4	4-11 bits

5.3.42. ImageMirrorMode

Definition	Value	Explanation
HORIZONTAL_MIRROR	0	Horizontal mirroring
VERTICAL_MIRROR	1	Vertical mirroring

5.3.43. ActualBits

Definition	Value	Explanation
BITS_10	10	10 bits
BITS_12	12	12 bits
BITS_14	14	14 bits
BITS_16	16	16 bits

5.3.44. TimerSelectorEntry

Definition	Value	Explanation
TIMER1	1	Timer 1

5.3.45. TimerTriggerSourceEntry

定义	Value	Explanation
EXPOSURE_START	1	Signal to start exposure

5.3.46. CounterSelectorEntry

Definition	Value	Explanation
COUNTER1	1	Counter 1

5.3.47. CounterEventSourceEntry

Definition	Value	Explanation
FRAME_START	1	Frame start

5.3.48. CounterResetSourceEntry

Definition	Value	Explanation
OFF	0	Counter reset off
SOFTWARE	1	Software trigger
LINE0	2	Pin 0
LINE1	3	Pin 1
LINE2	4	Pin 2
LINE3	5	Pin 3

5.3.49. CounterResetActivationEntry

Definition	Value	Explanation
RISING_EDGE	1	Rising edge counter reset

5.4. Interface definition

5.4.1. DeviceManager

It is responsible for the management of camera, including enumerating device, getting device list, opening device, etc.

◆ Interface list

List<DeviceInfo> updateDeviceList (int timeout);	Enumerate devices on the same network segment (General enumeration)
List<DeviceInfo> updateAllDeviceList (int timeout);	Enumerate all devices (General enumeration)
void setOnUpdateDeviceListFinishedListener(OnUpdateDeviceListFinishedListener listener);	Set the listener of finished enumeration (enumeration without Root permission)
void startUpdateDeviceList(Context context, int timeout);	Start enumeration process (enumeration without Root permission)
void startUpdateAllDeviceList(Context context, int timeout);	Start enumeration process (enumeration without Root permission). This interface is used when U3V devices and GEV devices are used simultaneously and all GEV devices need to be enumerated
List<DeviceInfo> getDeviceInfoList ();	Get the list of devices that have been enumerated
Device openDeviceBySN (String strSN, AccessMode accessMode);	Open the device by serial number
Device openDeviceByUserID (String strUserID, AccessMode accessMode);	Open the device by user ID number
Device openDeviceByIndex (int index, AccessMode accessMode);	Open the device by device index
Device openDeviceByIP (String strIP, AccessMode accessMode);	Open the device by IP address
Device openDeviceByMAC (String strMAC, AccessMode accessMode);	Open the device by MAC address

Sample code:

Reference to [Open the device](#).

◆ Interface description

➤ updateDeviceList

Declarations:

```
List<DeviceInfo> DeviceManager.updateDeviceList (int timeout);
```

Descriptions:

For non-GEV cameras, enumerate all devices. For GEV cameras, enumerate devices on the same network segment.

Parameters:

[in] timeout Enumeration timeout [0, 0x7fffffff], unit: ms

Returns:

The list of device information: **List<DeviceInfo>**. See [Enumeration device](#) for the information details that DeviceInfo contains.

Exceptions:

If getting the information of all devices unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **updateAllDeviceList**

Declarations:

List<DeviceInfo> DeviceManager.updateAllDeviceList (int timeout);

Descriptions:

For non-GEV cameras, enumerate all devices. For GEV cameras, enumerate network-wide devices.

Parameters:

[in] timeout Enumeration timeout [0, 0x7fffffff], unit: ms

Returns:

The list of device information: List<DeviceInfo>. See [Enumeration device](#) for the information details that DeviceInfo contains.

Exceptions:

If getting the information of all devices unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **setOnUpdateDeviceListFinishedListener**

Declarations:

```
void DeviceManager.setOnUpdateDeviceListFinishedListener(  
    OnUpdateDeviceListFinishedListener  
    listener);
```

Descriptions:

Set the listener of finished enumeration, mainly used with startUpdateDeviceList or startUpdateDeviceList interface

Parameters:

[in] listener The listener of finished enumeration

Exceptions:

If setting the listener of finished enumeration unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **startUpdateDeviceList**

Declarations:

```
void DeviceManager.startUpdateDeviceList (Context context, int timeout);
```

Descriptions:

Start enumeration process (enumeration without Root permission). This interface **can not be used** when using non-U3V camera. When U3V and GEV cameras are used simultaneously, only cameras that are in the same network segment as the host can be enumerated.

Note:

- 1) You **must** call the setOnUpdateDeviceListFinishedListener interface to set the listener of finished enumeration before calling this interface. See [section 1.4.2](#) for sample code details.
- 2) Calling multiple threads simultaneously is not supported.

Parameters:

[in] context Context objects. You can input "this" in MainActivity, or you can input it by getting the context through the getApplicationContext() interface
[in] timeout Enumeration timeout [0, 0x7fffffff], units: ms

Exceptions:

If starting enumeration process unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ startUpdateAllDeviceList

Declarations:

```
void DeviceManager.startUpdateAllDeviceList (Context context, int timeout);
```

Descriptions:

Start enumeration process (enumeration without Root permission). This interface **can not be used** when using non-U3V camera. For non-GEV cameras, enumerate all devices, for GEV cameras, enumerate network-wide devices.

Note:

- 1) You **must** call the setOnUpdateDeviceListFinishedListener interface to set the listener of finished enumeration before calling this interface.
- 2) Calling multiple threads simultaneously is not supported.

Parameters:

[in] context	Context objects, which can be got through the <code>getApplicationContext()</code> interface
[in] timeout	Enumeration timeout [0, 0x7fffffff], units: ms

Exceptions:

If starting enumeration process unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ getDeviceInfoList

Declarations:

```
DeviceManager.getDeviceInfoList();
```

Descriptions:

Get the list of device information

Returns:

The list of device information. The number of elements in the list is the number of devices enumerated. Elements in the list is DeviceInfo, see [Enumeration device](#).

➤ openDeviceBySN

Declarations:

Device DeviceManager.openDeviceBySN (String strSN, AccessMode accessMode);

Descriptions:

Open the device by serial number

Parameters:

[in] strSN	Serial number [String]
[in] accessMode	Open device mode, only for GEV cameras, see AccessMode

Returns:

Device object

Exceptions:

If opening the device unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **openDeviceByUserID**

Declarations:

Device DeviceManager.openDeviceByUserID (String strUserID, AccessMode accessMode);

Descriptions:

Open the device by user ID number

Parameters:

[in] UserID	User ID number [String]
[in] accessMode	Open device access mode, only for GEV cameras, see AccessMode

Returns:

Device object

Exceptions:

If opening the device unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **openDeviceByIndex**

Declarations:

Device DeviceManager.openDeviceByIndex (int index, AccessMode accessMode);

Descriptions:

Open the device by device index

Note:

The device index starts at 1

Parameters:

[in] index	Device index [1,2,3...The number of devices]
[in] accessMode	Open device mode, only for GEV cameras, see AccessMode

Returns:

Device object

Exceptions:

If opening the device unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ openDeviceByIP

Declarations:

Device DeviceManager.openDeviceByIP (String strIP, AccessMode accessMode);

Descriptions:

Open the GEV cameras by IP address

Parameters:

[in] strIP	The IP address of device [string]
[in] accessMode	Open device mode, only for GEV cameras, see AccessMode

Returns:

Device object

Exceptions:

If opening the device unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ openDeviceByMAC

Declarations:

Device DeviceManager.openDeviceByMAC (String strMAC, AccessMode accessMode);

Descriptions:

Open the GEV cameras by MAC address

Parameters:

[in] strMAC Device MAC address[string]
[in] accessMode Open device mode, only for GEV cameras, see [AccessMode](#)

Returns:

Device object

Exceptions:

If opening the device unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

5.4.2. Device

It is responsible for camera acquisition control, device close, configuration file import and export, and acquisition of device handles, etc.

◆ Interface list

int	getStreamChannelNum();	Get the number of stream channels supported by the current device
DataStream	getDataStream(int index)	Get DataStream object
void	streamOn ();	Start image acquisition, the camera starts transmitting image data
void	streamOff ();	Stop image acquisition, the camera stops transmitting image data
void	exportConfigFile (String filePath);	Export current configuration file
void	importConfigFile(String filePath, Boolean verify);	Import configuration file
void	closeDevice ();	Close device

Sample code:

Reference to [Open the device](#), [Acquisition control](#).

◆ Interface description

➤ **getStreamChannelNum**

Declarations:

```
int Device.getStreamChannelNum();
```

Descriptions:

Get the number of stream channels supported by the current device

Returns:

The number of stream channels

Note: Currently, GEV cameras, USB3.0 and USB2.0 cameras do not support multi-stream channels.

➤ **getDataStream**

Declarations:

```
DataStream Device.getDataStream(int index);
```

Descriptions:

Get the number of stream channels supported by the current device

Parameters:

[in] index The numerical order of stream. Starting from 0, temporarily support 0

Returns:

[DataStream](#) object

➤ **streamOn**

Declarations:

```
void Device.streamOn();
```

Descriptions:

Start image acquisition, the camera starts acquiring and transmitting image data

Exceptions:

If starting acquisition unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **streamOff**

Declarations:

```
void Device.streamOff();
```

Descriptions:

Send a stop command, the camera stops transmitting image data

Exceptions:

If sending stop command unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **exportConfigFile**

Declarations:

```
void Device.exportConfigFile(String filePath);
```

Descriptions:

Export current configuration file

Parameters:

[in] filePath File path

Exceptions:

If exporting the current configuration file unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **importConfigFile**

Declarations:

```
void Device.importConfigFile(String filePath, boolean verify);
```

Descriptions:

Import configuration file

Parameters:

[in] filePath **File path**
[in] verify **Whether all imported values will be verified for consistency**

Exceptions:

If importing configuration file unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ closeDevice

Declarations:

```
void Device.closeDevice();
```

Descriptions:

Close the device

Note:

If you still want to use this camera after you close the device, please reopen it.

Exceptions:

If closing the device unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

5.4.3. DataStream

It is responsible for acquiring images, setting acquisition parameters, and getting the statistical data of the acquired images.

◆ Interface list

void	setAcquisitionBufferNumber (int bufferSize);	Set the number of image acquisition buffer
int	getAcquisitionBufferNumber();	Get the number of image acquisition buffer
RawImage	getRawImage (int timeout);	Get a Raw image data
Bitmap	getBitmap(ProcessParam processParam, FrameInfo frameInfo, int timeout);	Get an image and convert it to Bitmap to return, the image information will be returned via the FrameInfo type parameters
void	getImageBySurface (Surface surface,	Get an image and convert it, then display it on the SurfaceView widget, the image information will be

	ProcessParam processParam, FrameInfo frameInfo, String path, int timeout);	returned via the FrameInfo type parameters
void	flushQueue ();	Clear device library image buffer queue

Note: getRawImage/getBitmap/getImageBySurface are three image acquisition interfaces, an application can only select one of them to acquire images. **The interfaces cannot be used at the same time.**

Sample code:

Reference to [Acquisition control](#).

◆ Interface description

➤ setAcquisitionBufferNumber

Declarations:

```
void DataStream.setAcquisitionBufferNumber(int bufferNumber);
```

Descriptions:

Set the number of the acquisition buffer

Parameters:

[in] bufferNumber **The number of the acquisition buffer, range:[1, 0x7ffffff]**

Exceptions:

If setting the number of the acquisition buffer unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ getAcquisitionBufferNumber

Declarations:

```
int DataStream.getAcquisitionBufferNumber();
```

Descriptions:

Get the number of the acquisition buffer

Parameters:

None

Returns:

The number of the acquisition buffer

Exceptions:

If getting the number of the acquisition buffer unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **getRawImage**

Declarations:

```
RawImage DataStream.getRawImage(int timeout);
```

Descriptions:

Get a Raw image data

Parameters:

[in] timeout **Get timeout [0, 0x7fffffff]**

Returns:

RawImage object

Exceptions:

- 1) If no image is acquired, timeout exception is thrown.
- 2) If the interface fails to execute, other exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **getBitmap**

Declarations:

```
Bitmap DataStream.getBitmap(ProcessParam processParam,  
                             FrameInfo frameInfo,  
                             int timeout);
```

Descriptions:

Get an image and convert it to Bitmap

Parameters:

[in] processParam	ProcessParam class object. It contains parameters related to image conversion and image quality improvement
[in] frameInfo	Image frame information, see section 3.5.2 for details
[in] timeout	Get timeout [0, 0x7fffffff]

Returns:

Bitmap object

Exceptions:

- 1) If the acquired image is an incomplete frame, null is returned.
- 2) If no image is acquired, timeout exception is thrown.
- 3) If the interface fails to execute, other exception is thrown. See [section 3.8.2](#) for exception type details.

➤ getImageBySurface

Declarations:

```
void DataStream.getImageBySurface(Surface surface,
                                   ProcessParam processParam,
                                   FrameInfo frameInfo,
                                   String path,
                                   int timeout);
```

Descriptions:

Get an image and convert the image format, then display it on the SurfaceView widget, the image information will be returned via the FrameInfo type parameters.

Parameters:

[in] Surface	The parameter is Surface type, imports the Surface object corresponding to the SurfaceView widget that displays the image
[in] processParam	The parameter is ProcessParam type, contains parameters related to image conversion and image quality improvement
[out] frameInfo	Image frame information, see section 3.5.3 for details
[in] path	Save the image path and name, only Raw images can be saved
[in] timeout	Get timeout[0, 0x7fffffff]

Returns:

None

Exceptions:

- 1) If the acquired image is an incomplete frame, the image will not be processed, displayed, and saved.
- 2) If no image is acquired, timeout exception is thrown.
- 3) If the interface fails to execute, other exception is thrown. See [section 3.8.2](#) for exception type details.

➤ flushQueue

Declarations:

```
void DataStream.flushQueue();
```

Descriptions:

Clear the camera acquisition buffer queue

Exceptions:

If clearing the camera acquisition buffer queue unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

5.4.4. RawImage

It is responsible for some operations about the Raw image.

◆ Interface list

RGBImage	convertToRGB(BayerConvertType type, ValidBit validBit, boolean flip);	Convert to RGB image format
ARGBImage	convertToARGB(BayerConvertType type, ValidBit validBit, boolean flip, byte alpha);	Convert to ARGB image format
RawImage	mirror(ImageMirrorMode mirrorMode, ValidBit validBit);	Mirror function
byte[]	getData();	Get the data of raw image
void	save(String path);	Save the data of raw image
int	getStatus();	Get the status of raw image
int	getWidth();	Get the width of raw image
int	getHeight();	Get the height of raw image
int	getOffsetX();	Get the OffsetX of the raw image
int	getOffsetY();	Get the OffsetY of raw image
int	getPixelFormat();	Get the pixel format of image
long	getFrameID();	Get frame ID
long	getTimestamp();	Get timestamp

Sample code:

Reference to Image acquisition and processing [getRawImage method](#).

◆ Interface description

➤ convertToRGB

Declarations:

RGBImage RawImage.convertToRGB ([BayerConvertType](#) type,
[ValidBit](#) validBit,
boolean flip);

Descriptions:

Convert to RGB format image by interpolation

Parameters:

[in] type	Converted interpolation, reference to BayerConvertType
[in] validBit	Valid bit selection that converting a non-8-bit Raw image to an 8-bit Raw image, reference to ValidBit
[in] flip	Whether the RGB image of output is flipped

Returns:

RGBImage object

Exceptions:

If converting unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ convertToARGB

Declarations:

ARGBImage RawImage.convertToARGB ([BayerConvertType](#) type,
[ValidBit](#) validBit,
boolean flip,
byte alpha);

Descriptions:

Convert to ARGB format image by interpolation

Parameters:

[in] type	Converted interpolation, reference to BayerConvertType
-----------	--

[in] validBit Valid bit. Valid bit selection that converting a non-8-bit Raw image to an 8-bit Raw image, reference to [ValidBit](#)
[in] flip Whether the RGB image of output is flipped
[in] alpha Alpha channel value, generally import 0xFF

Returns

ARGBImage object

Exceptions:

If converting unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ mirror

Declarations:

```
RawImage RawImage.mirror(ImageMirrorMode mirrorMode,
                          ValidBit validBit);
```

Descriptions:

Image mirror supports horizontal mirror mode and vertical mirror mode.

Note:

Non-8-bit Raw format images are converted to 8-bit Raw images first, and then perform mirror processing.

Parameters:

[in] mirrorMode Mirror mode, reference to [ImageMirrorMode](#)
[in] validBit Valid bit, valid bit selection that converting a non-8-bit Raw image to an 8-bit Raw image, reference to [ValidBit](#)

Returns:

ARGBImage object

Exceptions:

If mirror processing is unsuccessful, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ getData

Declarations:


```
byte[] RawImage.getData();
```

Descriptions:

Get raw data

Returns:

Raw data

➤ save**Declarations:**

```
void RawImage.save(String filePath);
```

Descriptions:

Save raw image data

Parameters:

[in] filePath **File path**

Returns:

None

Exceptions:

If saving raw images unsuccessfully, an exception is thrown.

➤ getStatus**Declarations:**

```
int RawImage.getStatus();
```

Descriptions:

Get the status of raw image

Returns:

Get the status of raw image, identify if the image is complete. Reference [FrameStatus](#) for data type

➤ **getWidth**

Declarations:

```
int RawImage.getWidth();
```

Descriptions:

Get the width of raw image

Returns:

The width of raw image

➤ **getHeight**

Declarations:

```
int RawImage.getHeight();
```

Descriptions:

Get the height of raw image

Returns:

The height of raw image

➤ **getOffsetX**

Declarations:

```
int RawImage.getOffsetX();
```

Descriptions:

Get the OffsetX of raw image

Returns:

Raw image OffsetX

➤ **getOffsetY**

Declarations:

```
int RawImage.getOffsetY();
```

Descriptions:

Get the OffsetY of raw image

Returns:

Raw image OffsetY

➤ **getPixelFormat**

Declarations:

int RawImage.getPixelFormat();

Descriptions:

Get the pixel format of image

Returns:

Pixel format

➤ **getFrameID**

Declarations:

long RawImage.getFrameID();

Descriptions:

Get frame ID

Returns:

Frame ID

➤ **getTimestamp**

Declarations:

long RawImage.getTimestamp();

Descriptions:

Get timestamp

Returns:

Timestamp

5.4.5. RGBImage

It is responsible for some operations about the RGB image.

◆ Interface list

void	imageImprovement (long colorCorrectionParam, ByteBuffer gammaLut, ByteBuffer contrastLut);	Image quality improvement
byte[]	getData();	Get the data of RGB image
ARGBImage	convertToARGB(byte alpha);	Convert to ARGB data
int	getWidth();	Get the image width
int	getHeight();	Get the image height
int	getOffsetX();	Get the image OffsetX
int	getOffsetY();	Get the image OffsetY
long	getFrameID();	Get frame ID
long	getTimestamp();	Get timestamp

Sample code:

Reference to image acquisition and processing [Convert Raw images to RGB images](#).

◆ Interface description

➤ imageImprovement

Declarations:

```
void RGBImage. imageImprovement (long colorCorrectionParam,
                                   ByteBuffer gammaLut,
                                   ByteBuffer contrastLut);
```

Descriptions:

Image quality improvement

Parameters:

[in] colorCorrectionParam	Color correction parameter must be read from the camera
[in] gammaLut	Gamma LUT. When it is null, no correlative processing is performed
[in] contrastLut	Contrast LUT. When it is null, no correlative processing is performed

Exceptions:

- 1) If colorCorrectionParam is 0 and gamma LUT and contrastLut are null, the function does not perform any processing.
- 2) If there is an error in processing, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **convertToARGB**

Declarations:

ARGBImage RGBImage.convertToARGB (byte alpha);

Descriptions:

Convert to ARGB format image by interpolation

Parameters:

[in] alpha Alpha channel value, generally import 0xFF

Returns:

ARGBImage object

Exceptions:

If converting unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details.

➤ **getData**

Declarations:

byte[] RGBImage.getData();

Descriptions:

Get RGB image data

Returns:

RGB image data

➤ **getStatus**

Declarations:

int RGBImage.getStatus();

Descriptions:

Get image status, identify if the image is complete

Returns:

Image status, reference [FrameStatus](#) for data type

➤ **getWidth**

Declarations:

```
int RGBImage.getWidth();
```

Descriptions:

Get the image width

Returns:

Image width

➤ **getHeight**

Declarations:

```
int RGBImage.getHeight();
```

Descriptions:

Get the image height

Returns:

Image height

➤ **getOffsetX**

Declarations:

```
int RGBImage.getOffsetX();
```

Descriptions:

Get the image OffsetX

Returns:

Image OffsetX

➤ **getOffsetY**

Declarations:

```
int RGBImage.getOffsetY();
```

Descriptions:

Get the image OffsetY

Returns:

Raw image OffsetY

➤ **getFrameID**

Declarations:

```
long RGBImage.getFrameID();
```

Descriptions:

Get frame ID

Returns:

Frame ID

➤ **getTimestamp**

Declarations:

```
long RGBImage.getTimestamp();
```

Descriptions:

Get timestamp

Returns:

Timestamp

5.4.6. ARGBImage

It is responsible for some operations about ARGB image.

◆ **Interface list**

void	imageImprovement (long colorCorrectionParam, ByteBuffer gammaLut, ByteBuffer contrastLut);	Image quality improvement
int[]	getData();	Get the data of ARGB image
Bitmap	getBitmap();	Get Bitmap
int	getWidth();	Get the image width
int	getHeight();	Get the image height
int	getOffsetX();	Get the image OffsetX
int	getOffsetY();	Get the image OffsetY
long	getFrameID();	Get frame ID
long	getTimestamp();	Get timestamp

Sample code:

Reference to Image acquisition and processing [Convert Raw images to ARGB images](#).

◆ Interface description

➤ imageImprovement

Declarations:

```
void ARGBImage.imageImprovement (long colorCorrectionParam,
                                   ByteBuffer gammaLut,
                                   ByteBuffer contrastLut);
```

Descriptions:

Image quality improvement

Parameters:

[in] colorCorrectionParam	Color correction parameter must be read from the camera
[in] gammaLut	Gamma LUT, when it is null, no correlative processing is performed
[in] contrastLut	Contrast LUT, when it is null, no correlative processing is performed

Exceptions:

- 1) If colorCorrectionParam is 0, and gamma LUT and contrastLut are null, the function does not perform any processing
- 2) If there is an error in processing, an exception is thrown. See [section 3.8.2](#) for exception type details

➤ getBitmap

Declarations:

Bitmap ARGBImage.getBitmap()

Descriptions:

Get Bitmap from ARGBImage

Returns:

Bitmap object

➤ **getData**

Declarations:

int[] ARGBImage.getData();

Descriptions:

Get ARGB data

Returns:

ARGB data

➤ **getStatus**

Declarations:

int ARGBImage.getStatus();

Descriptions:

Get image status, identify if the image is complete.

Returns:

Image status, reference [FrameStatus](#) for data type

➤ **getWidth**

Declarations:

int ARGBImage.getWidth();

Descriptions:

Get the image width

Returns:

Image width

➤ getHeight**Declarations:**

```
int ARGBImage.getHeight();
```

Descriptions:

Get the image height

Returns:

Image height

➤ getOffsetX**Declarations:**

```
int ARGBImage.getOffsetX();
```

Descriptions:

Get the image OffsetX

Returns:

Image OffsetX

➤ getOffsetY**Declarations:**

```
int ARGBImage.getOffsetY();
```

Descriptions:

Get the image OffsetY

Returns:

Image OffsetY

➤ **getFrameID**

Declarations:

```
long ARGBImage.getFrameID();
```

Descriptions:

Get frame ID

Returns:

Frame ID

➤ **getTimestamp**

Declarations:

```
long ARGBImage.getTimestamp();
```

Descriptions:

Get timestamp

Returns:

Timestamp

5.4.7. ByteBuffer

It is responsible for some operations about the ByteBuffer class, mainly responsible for the storage of data such as LUT, and pass the data to this interface library, also encapsulates the operation of reading and storing data from a file.

◆ **Interface list**

ByteBuffer();	Non-parameter constructor
ByteBuffer(int length);	The constructor of inputting Buffer size
ByteBuffer(byte[] bytes);	The constructor of inputting byte array
ByteBuffer(String path);	The constructor of inputting file path, reads the data from the file
byte[] getData();	Get data
int getLength();	Get the length of data
void save(String path);	Save the data to file

◆ **Interface description**

➤ **getData**

Declarations:

```
byte[] ByteBuffer.getData();
```

Descriptions:

Return the data of the ByteBuffer object

Returns:

Data (byte array)

➤ **getLength**

Declarations:

```
int ByteBuffer.getLength();
```

Descriptions:

Return the length of the ByteBuffer object

Returns:

The length of data

➤ **save**

Declarations:

```
void ByteBuffer.save(String filePath);
```

Descriptions:

Save data

Parameters:

[in] filePath `File path`

Returns:

None

Exceptions:

If saving the file path unsuccessfully, an exception is thrown.

5.4.8. Utility

It is responsible for some operations about gamma and contrast parameters.

◆ Interface list

ByteBuffer	getGammaLut(double gamma);	Get the gamma LUT by gamma value
ByteBuffer	getContrastLut(int contrast);	Get the contrast LUT by contrast value

Sample code:

Reference to [Image quality improvement](#).

◆ Interface description

➤ getGammaLut

Declarations:

ByteBuffer Utility. getGammaLut(double gamma)

Descriptions:

Get the gamma LUT by gamma value

Parameters:

[in] gamma Int or float, range [0.1, 10.0]

Returns:

Buffer object of gamma LUT

Exceptions:

If getting the gamma LUT unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details

➤ getContrastLut

Declarations:

ByteBuffer Utility. getContrastLut(int contrast);

Descriptions:

Get the contrast LUT by contrast value

Parameters:

[in] contrast Int, range [-50, 100]

Returns:

Contrast LUT

Exceptions:

If getting the contrast LUT unsuccessfully, an exception is thrown. See [section 3.8.2](#) for exception type details

6. Revision History

No.	Version	Changes	Date
1	V1.0.0	Initial release	2019-05-14