October 2020

API references for Camera Remote SDK

Camera Remote SDK API Reference

SONY

First edition (February 2020)

Preface

About this document

The purpose of this document is to list the API specifications for the Camera Remote SDK provided by Sony Imaging Products & Solutions Inc.

Document conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in IETF RFC 2119.

http://www.ietf.org/rfc/rfc2119.txt

For information regarding the latest Camera Remote SDK updates, go to the web site at

http://www.sony.net/CameraRemoteSDK/

Document history

Change history		
2020-02-06	Version 1.00.00	First version
2020-06-18	Version 1.00.01	Just SDK version proceeded with bug fix (no change in the API doc.)
2020-07-16	Version 1.01.00	Some of DeviceProperties and Property values added.
2020-07-28	Version 1.02.00	Supporting products is updated. Some of DeviceProperties and Property values added.
2020-08-03	Version 1.02.00	Providing package is updated.
2020-09-15	Version 1.02.00	Supporting products is updated.
2020-10-15	Version 1.02.01	Just SDK version proceeded with bug fix (no change in the API doc.) Windows version only.
2020-10-15	Version 1.02.01	Explanation of Focus_Magnifier_Setting is updated in CrDeviceProperty and added in Tips/Trouble Shooting.

Contents

Introduction

The purpose of this document is to describe the API specifications and information about how to access camera functions and the procedure to establish connection to use the APIs for the Camera Remote SDK.

Version, Serial Number, Providing Package

Version

The Camera Remote SDK itself has one version, the app may check this version and change its behavior accordingly.

Camera Remote SDK version

Camera Remote SDK has its version defined by its specifying functions. The version will be changed if an API is added or deleted. The version also will be changed if a supporting function in any APIs is changed. The Camera Remote SDK version can be obtained by the "GetSDKVersion" API. For details, please see the "GetSDKVersion" API specification.

Serial number

The Camera Remote SDK itself has a serial number, the app may check this serial number.

Camera Remote SDK serial number

Camera Remote SDK has its serial number. The Camera Remote SDK serial number can be obtained by the "GetSDKSerial" API. For details, please see the "GetSDKSerial" API specification.

Providing Package

Camera Remote SDK has following packages.

- Camera Remote SDK for Windows

- Camera Remote SDK for Linux 64bit (ARMv8)

- Camera Remote SDK for Linux 32bit (ARMv7)

- Camera Remote SDK for Linux 64bit (x86)

Supporting conditions

Even if the support conditions below are satisfied, it does not guarantee proper operation in all environments.

Supporting products and Help Guide URLs

Functions and parameters that are not supported by your camera cannot be used even if they are described in the API specification.

Please update each camera to the latest System Software (Firmware) before use.

- ILCE-7RM4	https://helpguide.sony.net/ilc/1930/v1/en/index.html
- ILCE-9M2	https://helpguide.sony.net/ilc/1960/v1/en/index.html
- ILCE-7SM3	https://helpguide.sony.net/ilc/2010/v1/en/index.html
- ILCE-7C	https://helpguide.sony.net/ilc/2020/v1/en/index.html (It will be available after product launch.)

Supporting physical layer

USB

Supporting OS

- Camera Remote SDK for Windows

  Checked with the environment on "Windows 8.1 64bit", "Windows 10 64bit"

- Camera Remote SDK for Linux 64bit (ARMv8)

  Checked with the environment below.

No.	Hardware	CPU	OS
1	Jetson Nano Developer Kit B01	ARMv8 Cortex-A57	Ubuntu 18.04.4 LTS (GNU/Linux 4.9.140-tegra aarch64)
2	Raspberry Pi4 Model B (4GB)	ARMv8 Cortex-A72	Raspberry Pi OS (64 bit) beta test version

- Camera Remote SDK for Linux 32bit (ARMv7)

  Checked with the environment below.

No.	Hardware	CPU	OS
1	Raspbery Pi2 Model B V1.1 (Broadcom BCM2836)	ARMv7 Cortex-A7	Raspberry Pi OS (32-bit) with desktop (Version: May 2020)

- Camera Remote SDK for Linux 64bit (x86)

  Checked with the environment on “Ubuntu 18.04 LTS”, “Ubuntu 20.04 LTS”

Even if the support conditions are satisfied, it does not guarantee proper operation in all environments.

Environment Setup

Change the USB Bulk Transfer Rate

USB Bulk Transfer Rate should be changed to 150. The way to set it depends on the OS.

Raspberry Pi OS

Open /etc/rc.local with an editor.

Add the command below at the end of the file before "exit 0" to modify Bulk Transfer Rate configuration file.

```
Add this command:
sudo sh -c 'echo 150 > /sys/module/usbcore/parameters/usbfs_memory_mb'


exit 0
```

Save & Close the file and reboot. Make sure that "150" is written in the configuration file.

```
$ cat /sys/module/usbcore/parameters/usbfs_memory_mb
150
```

Ubuntu (for Embedded)

Open /boot/extlinux/extlinux.conf with an editor.

Change "APPEND ${cbootargs} quiet" to the command below.

```
Before:
APPEND ${cbootargs} quiet
```

```
After:
APPEND ${cbootargs} usbcore.usbfs_memory_mb=150 usbcore.autosuspend=-1
```

Save & Close the file and reboot. Make sure that "150" is written in the configuration file.

```
$ cat /sys/module/usbcore/parameters/usbfs_memory_mb

150
```

Ubuntu (for x86)

Open /etc/default/grub with an editor.

Change “quiet splash” to the command below.

```
Before:

GRUB_CMDLINE_LINUX_DEFAULT="quiet splash“
```

```
After:

GRUB_CMDLINE_LINUX_DEFAULT="quiet splash usbcore.usbfs_memory_mb=150"
```

```
sudo update-grub
```

Save & Close the file and reboot. Make sure that “150” is written in the configuration file.

```
$ cat /sys/module/usbcore/parameters/usbfs_memory_mb

150
```

API list

APIs	Outline
Init	Initialize the Camera Remote SDK for use.
Release	Terminate the Camera Remote SDK.
EnumCameraObjects	Make a list of corresponding camera for the Camera Remote SDK.
CreateCameraObjectInfo	Create an ICrCameraObjectInfo object represents a Camera.
Connect	Connect to a Camera using a ICrCameraObjectInfo object before manipulation..
Disconnect	Disconnect from the Camera after use.
ReleaseDevice	Remove resources allocated with the Connect function.
GetDeviceProperties	Read camera settings.
ReleaseDeviceProperties	Release the CrDeviceProperty objects allocated by
SetDeviceProperty	Set camera settings.
SendCommand	Send control command.
GetLiveViewImage	Read the latest live-view image data from the Camera into the memory of the current machine.
GetLiveViewImageInfo	This function returns the size of the live-view image.
GetLiveViewProperties	Get live view properties from the camera.
ReleaseLiveViewProperties	Release the CrLiveViewProperty objects allocated by
GetDeviceSetting	This function returns the value of settings in the Camera Remote
SetDeviceSetting	This function modifies the value of settings in the Camera Remote
SetSaveInfo	This function modifies settings for saving pictures
GetSDKVersion	Get SDK version number.
GetSDKSerial	Get SDK serial number.

Function list

Functions	DeviceProperty Code / Command Id	ILCE-7RM4	ILCE-9M2	ILCE-7SM3	ILCE-7C
Shutter Half Release	CrDeviceProperty_S1	○	○	○	○
Shutter Release	CrCommandId_Release	○	○	○	○
AELock Indication	CrDeviceProperty_AEL	○	○	○	○
FEL Lock Indication	CrDeviceProperty_FEL	○	○	○	○
AWBLock Indication	CrDeviceProperty_AWBL	○	○	○	○
F-Number	CrDeviceProperty_FNumber	○	○	○	○
Exposure Bias Compensation	CrDeviceProperty_ExposureBiasCompensation	○	○	○	○
Shutter Speed	CrDeviceProperty_ShutterSpeed	○	○	○	○
ISO Sensitivity	CrDeviceProperty_IsoSensitivity	○	○	○	○
Focus Area	CrDeviceProperty_FocusArea	○	○	○	○
Exposure Program Mode	CrDeviceProperty_ExposureProgramMode	○	○	○	○
Compress File Format(Still)	CrDeviceProperty_CompressionFileFormatStill	×	×	○	○
File Format(Still)	CrDeviceProperty_FileType	○	○	○	○
JPEG Quality	CrDeviceProperty_JpegQuality	○	○	○	○
White Balance	CrDeviceProperty_WhiteBalance	○	○	○	○
Focus Mode	CrDeviceProperty_FocusMode	○	○	○	○
Exposure Metering Mode	CrDeviceProperty_MeteringMode	○	○	○	○
Flash Mode	CrDeviceProperty_FlashMode	○	○	○	○
Flash Compensation	CrDeviceProperty_FlashCompensation	○	○	○	○
Wireless Flash Setting	CrDeviceProperty_WirelessFlash	○	○	○	○
Red Eye Reduction	CrDeviceProperty_RedEyeReduction	○	○	○	○
Still Capture Mode	CrDeviceProperty_DriveMode	○	○	○	○
Dynamic Range Optimizer	CrDeviceProperty_DRO	○	○	○	○
Image Size	CrDeviceProperty_ImageSize	○	○	○	○
Aspect Ratio	CrDeviceProperty_AspectRatio	○	○	○	○
Picture Effect	CrDeviceProperty_PictureEffect	○	○	×	×
Color Temperature	CrDeviceProperty_Colortemp	○	○	○	○
Biaxial Fine Tuning A-B	CrDeviceProperty_ColorTuningAB	○	○	○	○
Biaxial Fine Tuning G-M	CrDeviceProperty_ColorTuningGM	○	○	○	○
Live View Display Effect	CrDeviceProperty_LiveViewDisplayEffect	○	○	○	○
Still Image Save Destination	CrDeviceProperty_StillImageStoreDestination	○	○	○	○
Position Key Setting	CrDeviceProperty_PriorityKeySettings	○	○	○	○

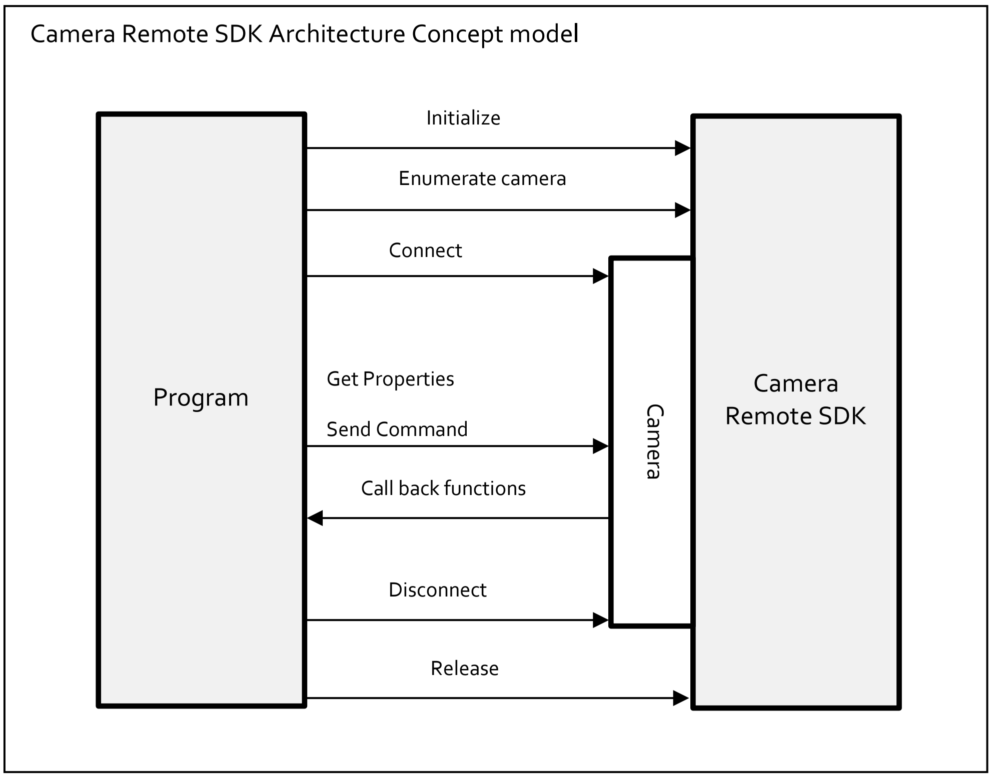
Focus Magnifier Setting	CrDeviceProperty_Focus_Magnifier_Setting	○	○	○	○
Focus Near/Far Setting	CrDeviceProperty_NearFar	○	○	○	○
Live View Image Quality	CrDeviceProperty_LiveView_Image_Quality	○	○	○	○
Interval REC Mode	CrDeviceProperty_Interval_Rec_Mode	○	○	○	○
Still Image Trans Size	CrDeviceProperty_Still_Image_Trans_Size	○	○	○	○
RAW+J PC Save Image	CrDeviceProperty_RAW_J_PC_Save_Image	○	○	○	○
Custom WB Capture Standby	CrDeviceProperty_CustomWB_Capture_Standby	○	○	○	○
Custom WB Capture Standby Cancel	CrDeviceProperty_CustomWB_Capture_Standby_Cancel	○	○	○	○
Custom WB Capture	CrDeviceProperty_CustomWB_Capture	○	○	○	○
Shooting File Info	CrDeviceProperty_SnapshotInfo	○	○	○	○
Battery Remaining	CrDeviceProperty_BatteryRemain	○	○	○	○
Battery Level Indicator	CrDeviceProperty_BatteryLevel	○	○	○	○
Movie Recording State	CrDeviceProperty_RecordingState	○	○	○	○
LiveView Status	CrDeviceProperty_LiveViewStatus	○	○	○	○
Focus Indication	CrDeviceProperty_FocusIndication	○	○	○	○
Media SLOT1 Status	CrDeviceProperty_MediaSLOT1_Status	○	○	○	○
Media SLOT1 Remaining number shots	CrDeviceProperty_MediaSLOT1_RemainingNumber	○	○	○	○
Media SLOT1 Remaining shooting time	CrDeviceProperty_MediaSLOT1_RemainingTime	○	○	○	○
Media SLOT1 Format Enable Status	CrDeviceProperty_MediaSLOT1_FormatEnableStatus	×	○	○	○
Media SLOT2 Status	CrDeviceProperty_MediaSLOT2_Status	○	○	○	×
Media SLOT2 Remaining number shots	CrDeviceProperty_MediaSLOT2_RemainingNumber	○	○	○	×
Media SLOT2 Remaining shooting time	CrDeviceProperty_MediaSLOT2_RemainingTime	○	○	○	×
Media SLOT2 Format Enable Status	CrDeviceProperty_MediaSLOT2_FormatEnableStatus	×	○	○	×
Media Format Progress Rate	CrDeviceProperty_Media_FormatProgressRate	×	○	○	○
Execute Format the Media	CrCommandId_MediaFormat	×	○	○	○
AF Area Position	CrDeviceProperty_AF_Area_Position	○	○	○	○
Zoom Scale	CrDeviceProperty_Zoom_Scale	○	○	○	○
Zoom Setting	CrDeviceProperty_Zoom_Setting	○	○	○	○
Zoom Operation	CrDeviceProperty_Zoom_Operation	○	○	○	○
File Format(Movie)	CrDeviceProperty_Movie_File_Format	○	○	○	○

Recording Setting(Movie)	CrDeviceProperty_Movie_Recording_Setting	○	○	○	○
Recording Frame Rate Setting(Movie)	CrDeviceProperty_Movie_Recording_FrameRateSetting	×	×	○	○
Interval REC Status	CrDeviceProperty_Interval_Rec_Status	○	○	○	○
Control Movie Rec button	CrCommandId_MovieRecord	○	○	○	○
Custom WB Execution State	CrDeviceProperty_CustomWB_Execution_State	○	○	○	○
Custom WB Capturable Area	CrDeviceProperty_CustomWB_Capturable_Area	○	○	○	○
Custom WB Capture Frame Size	CrDeviceProperty_CustomWB_Capture_Frame_Size	○	○	○	○
Custom WB Capture Operation Enable Status	CrDeviceProperty_CustomWB_Capture_Operation	○	○	○	○
Zoom Operation Enable Status	CrDeviceProperty_Zoom_Operation_Status	○	○	○	○
Zoom Bar Information	CrDeviceProperty_Zoom_Bar_Information	○	○	○	○
Zoom Type Status	CrDeviceProperty_Zoom_Type_Status	○	○	○	○

Operational Flow and Sequences

This section describes the basic operational flow of Camera Remote SDK.

At the beginning of all camera operations, Init() must be called to initialize Camera Remote SDK, and at the end of the operation, Release() must be called to release all resources.



EnumCameraObjects() enumerates connected cameras that can be connected with this Camera Remote SDK. The ICrEnumCameraObjectInfo object has the list of valid camera objects.

`ICrEnumCameraObjectInfo::GetCameraObjectInfo(CrInt32 index)` returns CrCameraObjectInfo specified by the parameter "index". With the CrCameraObjectInfo object, call the Connect() method to connect to the camera. Note that before calling Connect(), the IDeviceCallback function object needs to be prepared. The callback functions notify the status changes of the camera and the connection. When the connection established, OnConnected() is called. When the connection is disconnected, OnDisconnected() is called. When the camera status is changed, some other callback functions are called depending on the camera status, or warning / error messages are notified by the callback functions.

Connect() returns a CrDeviceHandle. The device handle is always used to operate the camera, for example to get or change properties, to capture image, to get live view images and so on. But just calling Connect() and receiving no error is not enough to know the timing the device is connected, and if the handle is validated. After the OnConnected() callback is called, the connection is established successfully, and the device handle is valid.

After using the camera, by calling the Disconnect() method with the device handle, the disconnect process starts. Similar to the Connect() method, when the OnDisconnected() callback function is called, the connection is disconnected successfully. You can call ReleaseaDevice() after you receive the OnDisconnected() call-back.

NOTE:

In this Camera Remote SDK, only one camera connection is guaranteed at the same time.

Initialize and Release Camera Remote SDK

To initialize Camera Remote SDK, call `SCRSDK::Init(0)`.

Init() needs one parameter, which must be zero.

In case of a memory allocation error or another fatal error, it returns false.

```
Example:

bool Init() {

            bool ret = SCRSDK::Init(0);

            if (!ret) {

                        // code to handle the error

                        return false;

            }
```

To terminate Camera Remote SDK, call `SCRSDK::Release()`. This function terminates all connections and releases the allocating resources. Note that the Release() function waits for the completion of the data transfer to be executed. When transferring huge amounts of data between the pc and the camera, this Release() function waits for the completion of the transfer. It is strongly recommended to call this method after confirming the disconnection of each device.

```
Example:

void Terminate() {

            SCRSDK::Release();
```

Currently, Release() always returns true.

Enumerate Cameras

EnumCameraObjects() enumerates all connectable cameras that are physically connected to the PC. Returned ICrEnumCameraObjectInfo has the list of the cameras. The ICrEnumCameraObjectInfo object is created in Camera Remote SDK, if no camera is found, the returned pEnum is NULL.

The member function GetCount() of ICrEnumCameraObjectInfo returns the number of the discovered cameras and GetCameraObject(index) returns the CrCameraObjectInfo object specified by the index parameter. Information of the discovered camera can be acquired through the object. The information varies depending on the connecting method. Connecting by USB allows you to acquire various information values (camera model name, product id, USB serial number, etc.).

To release ICrEnumCameraObjectInfo object, use the Release() function of the object.

```
Example:

void Enumerate() {

        CrError err = SCRSDK::EnumCameraObjects(&pEnum);

        if (pEnum == NULL) {

                // no cameras found

                return;

        }

        CrInt32u cntOfCamera = pEnum->GetCount(); // get number of cameras

        for (CrInt32u n = 0; n < cntOfCamera; n++) {

                ICrCameraObjectInfo *pobj = pEnum->GetCameraObjectInfo(n);

                // get connected camera information

        }

        :

        pEnum->Release(); // use Release() function of ICrEnumCameraObjectInfo
```

This enumeration function makes the list of "connectable" cameras. A Sony camera, which does not have PC remote control features or is not compatible with this Camera Remote SDK, is not listed. Refer to the supported model list of this Camera Remote SDK.

Note that ICrCameraObjectInfo *pobj in the sample code is the object owned by ICrEnumCameraObjectInfo. It means calling `ICrEnumCameraObjectInfo::Release()` frees the memory of ICrCameraObjectInfo that you get from the enumerator. It can no longer be accessed.

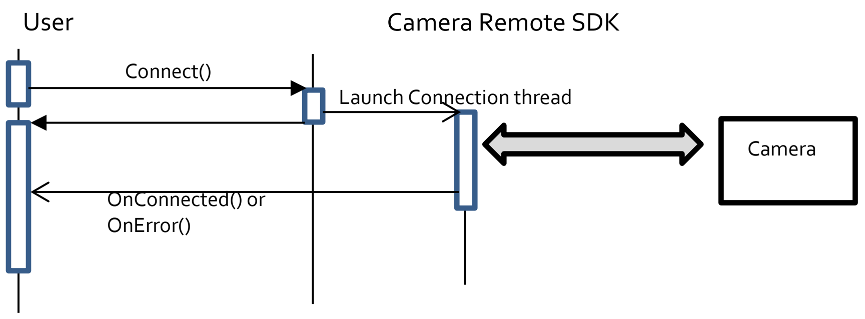
Connect a Camera

Using one of the enumerated ICrCameraObjectInfo, the camera can be connected with Camera Remote SDK by calling the Connect() function of the class. This function has three parameters. The first parameter ICrCameraObjectInfo * specifies the camera to connect to. The second parameter IDeviceCallback is a function object that is called back to notify the communication events from Camera Remote SDK. The caller must create the object instance before calling the Connect() function. The third parameter CrDeviceHandle * is returned with the connection handle from SDK and it must be set NULL before calling the Connect() function.

After the Connect() function, ICrCameraObjectInfo can be freed. There is no need to wait for OnConnected() or the OnError() callback function. It means you can delete the ICrEnumCameraObjectInfo object returned from the EnumCameraObject() function.

```
Example:

class MyDeviceCallback : public IDeviceCallBack {

             void OnConnected(DeviceConnectionVersion version) {

                                          DeviceConnectionVersion ver = version;

                                          // Program can use the device handle.

                          };

             :

bool ConnectCamera(ICrCameraObjectInfo *pcamera) {

             MyDeviceCallback *cb = new MyDeviceCallback();

             CrDeviceHandle            hDev = NULL;

             CrErr err = SCRSDK::Connect(pcamera, cb, &hDev);
```

As described at the top of this section, the connection process is executed asynchronously. Calling the Connect() function means that just the connection process is started. When the connection is established, the OnConnected() callback of ICrDeviceCallback is called.



The left vertical line indicates the user thread of your program, the center vertical line indicates API of Camera Remote SDK, and the right vertical line indicates the camera connection thread inside Camera Remote SDK.

Connect() returns an error where the function parameter is not valid. In the synchronous process in the Connect() function, it does not check for the device existence or the connectivity. It is checked in the Connection thread. If the camera is not found or if the camera is not compatible with the Camera Remote SDK, the OnError() callback function is called with an error id, CrError_Connect_Connect.

If the connection is established, the OnConnect() callback function is called with a parameter for connecting Remote Control Protocol Version.

In this Camera Remote SDK version, the parameter's value below is fixed.

Device_Connection_Version_RCP3 = 300

Because this version's Camera Remote SDK supports only the Remote-Control Protocol Version 3.

Disconnect a Camera

Call the Disconnect() function to disconnect the camera. The function needs one parameter for the DeviceHandle to disconnect.

Example:

```
void Disconnect(CrDeviceHandle handle) {

                SCRSDK::Disconnect(handle);
```

If the handle is not valid, Disconnect() returns an error.

Disconnect() is also an asynchronous process. The return from Disconnect() does not mean that the camera has been disconnected. At the time of the OnDisconnected() callback function is called, the camera has been disconnected from the Camera Remote SDK.

Get the Camera Properties

After the connection is established, camera property can be acquired by the GetDeviceProperties() function. This function has three parameters. The first one is the device handle that specifies the device, the second one is the pointer to CrDeviceProperty pointer that receives the acquired property list, and the third one receives the size of the CrDeviceProperty list.

The CrDeviceProperty returned from GetDeviceProperties() is allocated in Camera Remote SDK and the memory MUST be freed by calling ReleaseDeviceProperties() function.

```
Example:

void GetProperties(CrDeviceHandle handle) {

        CrDeviceProperty *pProperties;

        CrInt32u numofProperties = 0;

        SCRSDK::GetDeviceProperties(handle, &pProperties, &numofProperties);

        if (pProperties) {        // the property list is received successfully

                for (CrInt32u n = 0; n < numofProperties; n++) {

                        switch (pProperties[n].code) {

                        case CrDeviceProperty_FNumber:

                        // code to parse the properties...

                :

                }

                SCRSDK::ReleaseDeviceProperties(handle, pProperties);

        }
```

In the sample code above, for code simplification, the return value of GetDeviceProperties() is not checked, but it has to be checked. If the camera has already disconnected, it returns CrError_Invalid_Parameter. Additionally, in case of device property memory allocation error, it returns CrError_Generic_Unknown.

The content of the property list depends on the camera features. It is not expected that the all of properties are defined in enum of CrDevicePropertyCode in CrDevicePropty.h. Some properties defined in CrDevicePropertyCode will also be acquired by the GetLiveViewProperties() function as described in the following section.

This function does not communicate with the camera. This function returns the copy of the latest property list. The camera properties are updated automatically inside this Camera Remote SDK. In case of one or other properties are changed, Camera Remote SDK calls OnPropertyChanged(). Camera Remote SDK assumes that GetDeviceProperties() is called at the beginning of the camera operation, and when Camera Remote SDK calls the OnPropertyChanged() call back function. But calling the GetDeviceProperties() function in the OnPropertyChanged() callback function is not recommended, because the callback function is called on the thread that communicates with the camera. All callback functions are expected to return as soon as possible.

The following sample code is one of the references to get updated properties and to update the user interface items in Windows.

```
Example:

void MyDeviceCallback::OnConnected() {

             ::PostMessage(wnd, WM_APP_UPDATE_PROPERTIES, 0L, 0L);

}

void MyDeviceCallback::OnPropertyChanged() {

             ::PostMessage(wnd, WM_APP_UPDATE_PROPERTIES, 0L, 0L);

}


ON_MESSAGE(WM_APP_UPDATE_PROPERTIES, OnMessageUpdateProperties)


void CAppWnd::OnMessageUpdateProperties(WPARAM wp, LPARAM lp)

{

             GetProperties(handle);

             :  // update user interface items
```

Get the Live View Properties

Some camera properties cannot be acquired by the GetDeviceProperties() function. The properties that are defined in CrLiveViewPropertyCode are independent from the device property list, and must use the GetLiveViewProperties() function, because those properties are strongly related to the live view image.

The function interface and the usage are similar to GetDeviceProperties().

Similar to the device properties, the memory object returned from GetLiveViewProperties() must also be freed by ReleaseLiveViewProperties().

```
Example:

void GetLiveViewProperties(CrDeviceHandle handle) {

             CrLiveViewProperty *pProperties = NULL;

             CrInt32u numofProperties = 0;

             SCRSDK::GetLiveViewProperties(handle, &pProperties, &numofProperties);

             if (pProperties) {        // the property list is received successfully

                          for (CrInt32u n = 0; n < numofProperties; n++) {

                                       switch (pProperties[n].code) {

                                       case CrLiveViewProperty_AF_Area_Position:

                                       // code to parse the properties...

             :

                          }

                          SCRSDK::ReleaseLiveViewProperties(handle, pProperties);

             }
```

Device Properties and Live View Properties

CrDeviceProperty class and CrLiveViewProperty class store similar property values. The contents and the differences are explained in this section.

The CrDeviceProperty class has the following member variables shown below:

- code : Identify the content of the property.
- value Type : Specify the value variable type.
- enable Flag : Capability of the operation. Modifiable / Get Only / Invalid / Set Only
- current Value : Current property value. This value is defined as a 64bit variable.

If the property has a limited number of options, it has a list of the selectable options.

- value Size : Number of the selectable options.
- values : List of the selectable options.

The property code is defined in enum CrDevicePropertyCode in CrDeviceProperty.h. For example, CrDeviceProperty_FNumber is defined as 0x0100. The value type is CrDataType_UInt16. The current value is defined as a 64bit variable, but in this case only the highest 16bit is valid.

```
Example:

             switch (property->code) {

             case CrDevicePropty_FNumber:

                          CrInt16u     currentvalue = static_cast<CrInt16u>(property-
>currentValue);

                          :
```

If the enable flag is modifiable, the property can be acquired and can be set. To change the property value, refer to the SetDeviceProperty() function described in the next section. If the enable flag is Get Only, the property can be acquired and be referred to by GetDeviceProperties(), but cannot be changed.

Invalid means the property is invalid. This property must not be referred to or set. Set Only is also a very special case, as you see there is no "SetLiveViewProperty()" function. The properties you get via GetLiveViewProperties() are properties closely related to the live view feature, but in order to change the property you can use the SetDeviceProperty() function.

Depending on the camera status, this flag value changes. In case of CrDeviceProperty_FNumber, if the exposure mode of the camera is "M" or "A", this flag is modifiable, and in case of "P" or "S", this flag is Get Only.

If the property has selectable options, it has the list and the count of the list. Please note that the size is "Byte Size", not the count of the elements. Therefore, dividing by the size of the value type, the count of the elements can be calculated.

See the following reference pages to understand the property code and the type definitions.

```
Example:

            switch (property->code) {

            case CrDevicePropty_FNumber:

                        CrInt16u     currentvalue = static_cast<CrInt16u>(property->currentValue);

                        CrInt32u countofelement = property->valueSize / sizeof(CrInt16u);

                        CrInt16u *poptions = static_cast<CrInt16u*>property->values;

                        if (countofelement) {

                                    CrInt16u *elements = new CrInt16u [elementcount];

                                    for (CrInt32u n = 0; n < countofelement; n++) {

                                                elements[n] = *poptions++;
```

The CrLiveViewPropety class has similar members but there is "value size" to specify the memory size of current value, and there is no "selectable option" and its size field.

- code : Identify the content of the property.
- value Type : Specify the frame data type of value.
- enable Flag : Capability of the operation. Get Only
- value Size : Memory size in Bytes of Current property value.
- value : Current property value. This value is a memory block.

This value size is larger than CrDeviceProperty, because CrLiveViewProperty has the properties that represent coordination, regions or in some cases include the style. The definitions of the data type are described in the header file of "CrDeviceProperty.h" and the following reference section.

Because this CrLiveViewProperty class tells the information of the focus area, live view display magnification region, or custom white balance region, the API to get the properties from the camera is separated from GetDeviceProperties().

But note that to change those properties, the SetDeviceProperty() command must be used.

```
Example:

            switch (property->code) {

            case CrDeviceLiveViewProperty_AF_Area_Position:

                         CrFocusFrameInfo *pinfo

                        = static_cast<CrFocusFrameInfo *>(property->value);
```

Change the Camera Properties

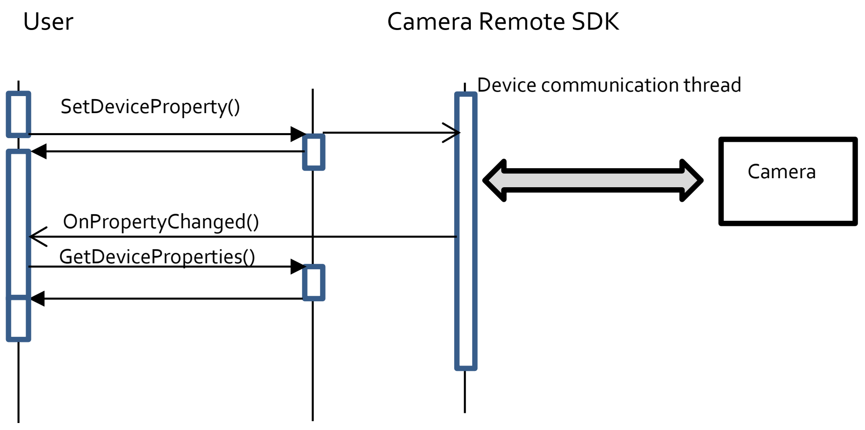
To change camera properties, for example F number, shutter speed, ISO and so on, send change property commands by using SetDeviceProperty(). There are two parameters, the first parameter is the device handle of the target camera, and the second parameter is the CrDeviceProperty class object. In this CrDeviceProperty object, only the code and value members are referred to in Camera Remote SDK.

If the value is invalid, the command is ignored, for example, where the out-of-range F number or setting F number in exposure mode is "S".

The combination of the code and the available value is described in API Reference section.

Note that this SetDeviceProperty() call is not synchronous. Once SetDeviceProprty() is called, the command is queued in the command queue in Camera Remote SDK and it is transmitted to camera at the appropriate time. It means that there is a short time lag between this function call and the camera's property change.

The properties in Camera Remote SDK are also not updated by the SetDeviceProperty() function. SDK keeps the property status of the camera. It is updated after the camera changes its status.



If the property is not changed because of the camera status, Camera Remote SDK does not notifies you of anything. It is recommended to set the 3- to 5-second timer in the user interface and try to get the property status to SDK and update the user interface state.

Send a Control Command

Some of the camera commands are implemented as "Control Command". For example, shutter release (fully pressing the shutter button), movie record and so on. In these cases, the SendCommand() function must be used. The interface of this function is much simpler than the device property case.

void SendCommnad(CrDeviceHandle device, CrInt32u commandId, CrCommandParam parm);

The first parameter specifies the device, the second parameter indexes the command id and the last parameter is ON (CrCommnadParam_Down) or OFF (CrCommandParam_Up). The Up and Down expresses the physical button action.

The following example shows how to capture images.

```
Example:
            SCRSDK::SendCommand(handle, CrCommandId_Release, CrCommandParam_Down);
```

This command initiates a human's action using the button; therefore, the button must be released (Up) once when you send "Down" command. If the camera's drive mode is in the continuous shooting mode, the camera captures continuously what it receives from the CrCommandParam_Down command until it receives CrCommandParam_Up.

This sample code shows the simplest way to press the shutter release button for one second.

```
Example:
            SCRSDK::SendCommand(handle, CrCommandId_Release, CrCommandParam_Down);
            Sleep(1000);
            SCRSDK::SendCommand(handle, CrCommandId_Release, CrCommandParam_Up);
```

This command sent by SendCommand() has a higher priority than other communication processes, getting device properties, and getting live view image data and so on, to make the response of camera quicker.

Get a Live View Image

Live view image is sent from the camera as a Jpeg image. The image size depends on the live view image quality of the camera setting and the image aspect mode.

The image is updated at a rate of 30 frames per second if the communication speed is good. The FPS becomes much lower when the communication bandwidth is narrow. The situations, where the communication quality is poor or where captured images are transmitted, result in corresponding to a lower live view FPS.

To receive live view image, a receive buffer needs to be prepared. The buffer size can be acquired by the GetLiveViewImageInfo() function. The first parameter is the device handle, and the second parameter is the pointer to CrImageInfo. CrImageInfo has the information related to width, height and the required buffer size. After getting the image buffer size, allocate the memory buffer for the image and call GetLiveViewImage().

```
Example:

            CrImageInfo *pInfo = new CrImageInfo();

            SCRSDK::GetLiveViewImageInfo(handle, pInfo);

            CrImageDataBlock *pLiveViewImage = new CrImageDataBlock();

            pLiveViewImage->frameNo = 0;

            pLiveViewImage->size = pInfo->bufferSize;

            pLiveViewImage->pData = new CrInt8u[pLiveViewImage->size];

            SCRSDK::GetLiveViewImage(handle, pLiveViewImage);
```

```
Example:

            SCRSDK::GetLiveViewImage(handle, pLiveViewImage);

            CrInt32u size = pLiveViewImage->GetImageSize();

            CrInt8u *pdata = pLiveViewImage->GetImageData();
```

CrImageInfo has the Jpeg image data and its size. GetImageData() returns the data pointer and GetImageSize() returns the data size.

This Jpeg image data starts from SOI marker (FF D8) and ends with EOI marker (FF D9). It can be displayed as it is by the graphic user interface using OpenGL, DirectDraw or another framework.

Example:

```
SCRSDK::GetLiveViewImage(handle, pLiveViewImage);

CrInt32u offset = *static_cast<CrInt32u*>(pLiveViewImage->pData);

CrInt32u size = *static_cast<CrInt32u*>(pLiveViewImage->pData + sizeof(CrInt32u));

CrInt8u *pJpegData = new CrInt8u[size];

memcpy(pJpegData, pLiveViewImage->pData + offset, size);
```

The image is updated inside Camera Remote SDK and one unique and an incremental number is given for the image that is transmitted from the camera. GetLiveViewImage() compares the frame number of the given CrImageDataBlock class object and the current frame number in the Camera Remote SDK. If the given number is smaller than the current number, a copy of the new image buffer is made of the given object and updates the frame number of the given object. If the number of the object is equal or larger than the number of the SDK, no copy is made, and it returns CrError_Frame_NotUpdated. Therefore, at the first call of GetLiveViewImage(), the frame number of CrImageDataBlock should be set to zero.

The size member of CrImageDataBlock is updated to the real image data size in GetLiveViewImage(). Where the buffer size of CrImageDataBlock is smaller than received image size, Camera Remote SDK also does not copy the buffer and returns CrError_Memory_Insufficient.

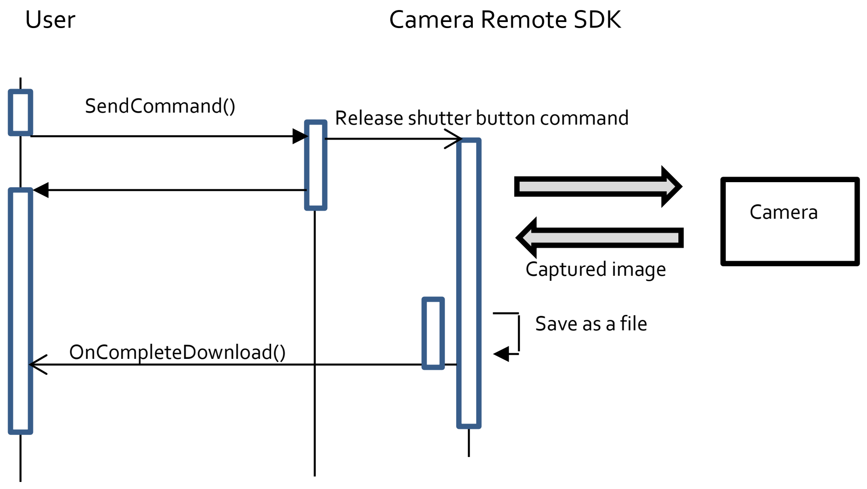
If the return value of the GetLiveViewImage() is CrWarning_Frame_NotUpdated, wait for a while and get the frame again. If the return value is CrError_Memory_Insufficient, get the image buffer size by GetLiveViewImageInfo() and reallocate the memory as the new size.

If GetLiveViewImage() returns CrError_Generic_Unknown, it means that there is an issue related to the data communication between the PC and Camera.

Capture an Image Sequence

Where the store image destination (CrDeviceProperty_StillImageStoreDestination) property is "PC" or "PC and Camera", the captured image is automatically transmitted to PC and stored in the storage of the PC by Camera Remote SDK.

This section explains the sequence of the storing captured images.



After Camera Remote SDK stored the image to a file, the OnCompleteDownload() callback function is called with the stored file path.

```
void OnCompleteDownload(CrChar *filename);
```

The store image folder can be set using the SetSaveInfo() function. The next section explains this process.

Change the Store Image Folder and the File Name

Camera Remote SDK has two modes to specify the image file name. One is "Auto Mode" and the other is "Manual Mode".

Auto Mode gives the image file name that is determined by the camera. In this case the naming rule of the camera is used. If the file name conflicts with an existing file, an additional number is appended after the file name like `DSC01234(1)`.JPG.

In Manual Mode, your program can specify the file name prefix and the start number. "ABCDE" as prefix and 100 as the start number makes the name from "ABCDE00100.JPG". To change the mode and the prefix and start number, use the SetSaveInfo() function. In this case, Camera Remote SDK finds a number that does not conflict with existing files and incrementally sets the file number like `ABCDE00100(1).JPG`.

The SetSaveInfo() function has four parameters. The first parameter specifies the device handle, the second parameter specifies the folder path to store, the third parameter specifies the file prefix string and the last parameter specifies the start number that is added to the file name.

To change to Auto Mode, set the null string (note that it means "", not null pointer) and give -1 as the start number.

```
Example:
            SCRSDK::SetSaveInfo(handle, L"C:¥Image", L"", -1);
```

Using Manual Mode and the specified prefix, set the string of the parameter. For example, to store the images in "C:\Image", set the string giving the "ABCDE" prefix and the sequential number from 00100.

```
Example:
            SCRSDK::SetSaveInfo(handle, L"C:¥Image", L"ABCDE", 100);
```

Camera Remote SDK works in Unicode, the folder path and the prefix must be set as Unicode string.

SDK Properties

Using SetDeviceSetting(), some behavior of Camera Remote SDK can be changed. The setting can be set for each device.

CrError SetDeviceSetting(CrDeviceHandle handle, CrInt32u key, CrInte32u value);

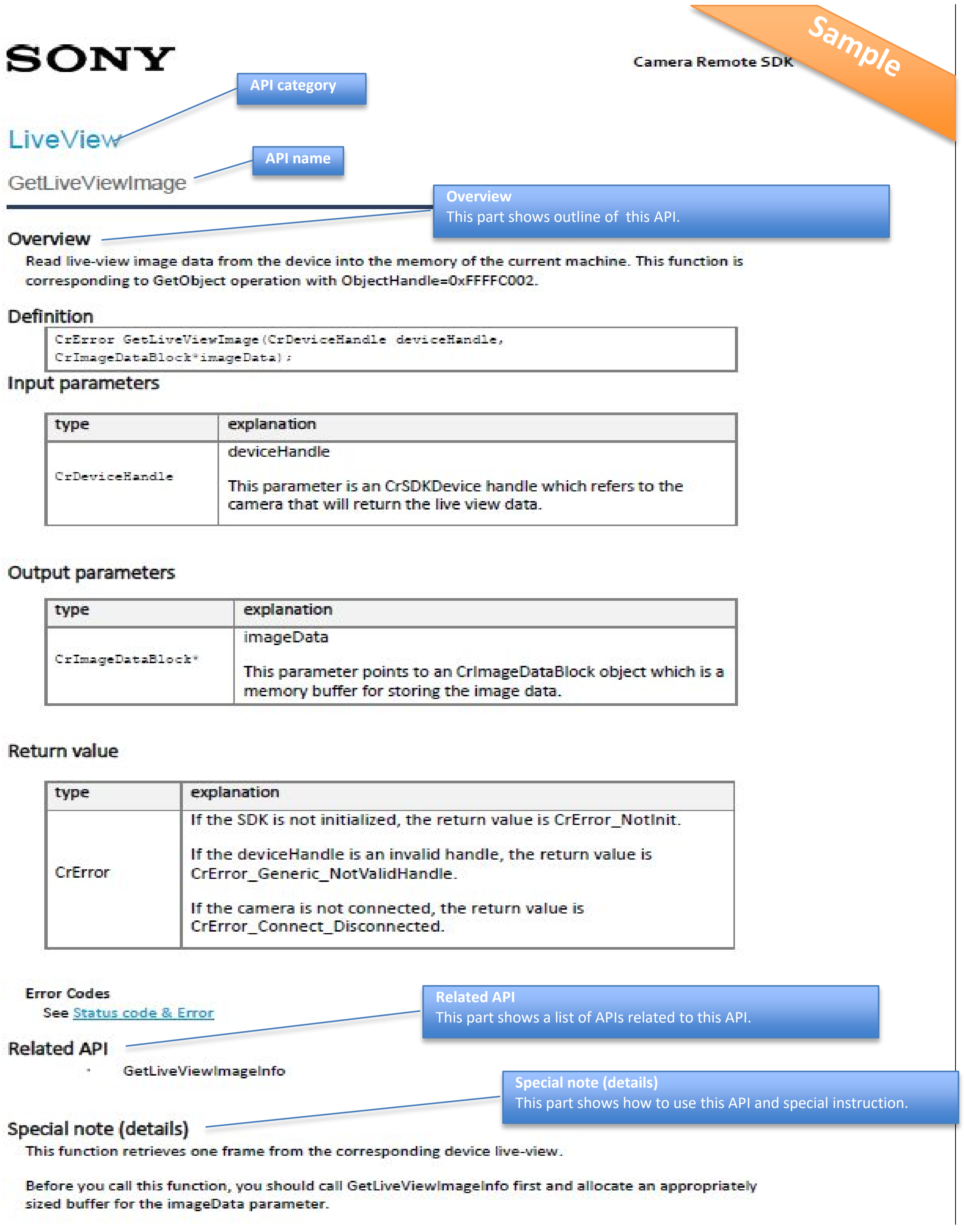
```
Example:
            SCRSDK::SetDeviceSetting(handle, Setting_Key_EnableLiveView, 0);
                         :
            SCRSDK::SetDeviceSetting(handle, Setting_Key_EnableLiveView, 1);
```

The code sample above disables and enables the live view feature. Set Zero to disable and set one to enable the feature.

In this version of Camera Remote SDK, only the Setting_Key_EnableLiveView setting can be set.

API Reference

This chapter provides the detailed API specification of Camera Remote SDK using the below format.


Sample
API category
API name
Overview
This part shows outline of this API.
Related API
This part shows a list of APIs related to this API.
Special note (details)
This part shows how to use this API and special instruction.

Camera Remote SDK

LiveView

GetLiveViewImage

Overview

Read live-view image data from the device into the memory of the current machine. This function is corresponding to GetObject operation with ObjectHandle=0xFFFFC002.

Definition

```
CrError GetLiveViewImage(CrDeviceHandle deviceHandle,
CrImageDataBlock*imageData);
```

Input parameters

type	explanation
CrDeviceHandle	deviceHandle This parameter is an CrSDKDevice handle which refers to the camera that will return the live view data.

Output parameters

type	explanation
CrImageDataBlock*	imageData This parameter points to an CrImageDataBlock object which is a memory buffer for storing the image data.

Return value

type	explanation
CrError	If the SDK is not initialized, the return value is CrError_NotInit. If the deviceHandle is an invalid handle, the return value is CrError_Generic_NotValidHandle. If the camera is not connected, the return value is CrError_Connect_Disconnected.

Error Codes

See Status code & Error

Related API

- GetLiveViewImageInfo

Special note (details)

This function retrieves one frame from the corresponding device live-view.

Before you call this function, you should call GetLiveViewImageInfo first and allocate an appropriately sized buffer for the imageData parameter.

Initialize

Init

Overview

Initialize the Camera Remote SDK for use. This function must be called before calling any other Camera Remote SDK function.

Definition

```
bool Init(CrInt32u logtype = 0)
```

Input Parameters

type	explanation
CrInt32u	Logtype. Only 0 is available in this version.

Return values

type	explanation
bool	Return parameter If initialize successfully, the result is true; otherwise, the result is false.

Related API

- Release

Special note (details)

None in particular

Release

Release

Overview

Terminate the Camera Remote SDK by deleting objects and releasing the memory used by the Camera Remote SDK. Use this function to clean up resources when the Camera Remote SDK is no longer required. Should be called after disconnecting all connected cameras and before your application close.

Definition

```
bool Release();
```

Input Parameters

Empty.

Return values

type	explanation
bool	Always returns true

Related API

- Init

Special note (details)

None in particular.

CameraObject

EnumCameraObjects

Overview

The API generates a list of “connectable” cameras. Even if a Sony camera is visible to the PC, if the camera doesn’t have PC remote control feature or if the camera doesn’t have compatibility with this version of Camera Remote SDK, the camera is not listed. Please refer the target model list of this Camera Remote SDK.

Definition

```
CrError EnumCameraObjects(ICrEnumCameraObjectInfo** ppEnumCameraObjectInfo,
CrInt8u timeInSec = 3);
```

Input parameters

type	explanation
`CrInt8u`	timeInSec This parameter is not supported with the current Camera Remote SDK.

Output parameters

type	explanation
`ICrEnumCameraObjectInfo**`	ppEnumCameraObjectInfo This is an input/output parameter. When this API returns, ppEnumCameraObjectInfo points an enumerator object to enumerate the connected cameras. If this pointer is null, no suitable camera devices were found. When the function returns successfully, the new object will be allocated within the function by the SDK. And because this pointer is overwritten in the SDK, calling EnumCameraObjects with unreleased memory object of this parameter will cause of leaking memory.

Return value

type	explanation
CrError	CrError_None on Success CrError_Init if the SDK is uninitialized CrError_Adaptor_HandlePlugin if any plugin modules are not found Other than errors above, see Status code & Error

Related API

- Connect
- ICrEnumCameraObjectInfo::Release

Special note (details)

This is a factory function. Release the list by calling ICrEnumCameraObjectInfo::Release interface function.

Enumerates all supported devices which are currently connected to the PC.

If no supported devices are found, ppEnumCameraObjectInfo remains nullptr.

If supported devices are discovered, ppEnumCameraObjectInfo points to the enumerator object. Their related information can be accessed through the ICrEnumCameraObjectInfo interface.

The information obtained through this API is required by the SDK Connect API.

CreateCameraObjectInfo

Overview

ICrCameraObjectInfo is an interface to detect a connectable camera that is connected via USB to the PC. It can be retrieved by ICrEnumCameraObjectInfo using GetCameraObjectInfo(), but can be created by calling CreateCameraObjectInfo(). This ICrCameraObjectInfo interface is used when the program connects a camera.

Definition

```
ICrCameraObjectInfo* CreateCameraObjectInfo(CrChar* name, CrChar* model,
CrInt16 usbPid, CrInt32u idType, CrInt32u idSize, CrInt8u* id, CrChar*
connecttypename, CrChar* adaptorname, CrChar* pairingnecessity);
```

Input parameters

type	explanation
`CrChar*`	name Not available.
`CrChar*`	model Null-terminated device model name string
`CrInt16`	usbPid Pid for usb devices
`CrInt32u`	idType For PTP_USB, this is CAMERAOBJECTID_TYPE_STRING
`CrInt32u`	idSize Size in bytes of the id buffer
`CrInt8u*`	id A buffer containing device information
`CrChar*`	connecttypename A char pointer which points to the null-terminated string of the connection type name of the camera. For PTP_USB, the string is "PTP-USB";
`CrChar*`	adaptorname A char pointer which points to the null-terminated string of the adapter name of the camera. For PTP_USB, the string is "Cr_PTP_USB";
`CrChar*`	reserved Call with NULL, because this parameter is not used.

All input parameter values are obtained from the EnumCameraObjects API. The user must decide how to preserve these values for use by the Connect API.

Output parameters

None

Return value

type	explanation
`ICrCameraObjectInfo*`	A pointer which points to a newly allocated ICrCameraObjectInfo interface object. The allocation is performed internally by the SDK. An object of this type is required when calling the Connect API.

Error Codes

See Status code & Error

Related API

- Connect
- EnumCameraObjects
- ICrCameraObjectInfo::Release

Special note (details)

This is a factory function that returns an ICrCameraObjectInfo* to an object allocated by the SDK. An ICrCameraObjectInfo is required to call the Connect API and connect to the corresponding device.

Remember to release the obtained ICrCameraObjectInfo by calling the ICrCameraObjectInfo::Release() interface function. Do not call delete manually.

Connection

Connect

Overview

This API attempts to connect to the camera device specified by the user.

This function is an asynchronous connection request. If this function returns without error, the asynchronous connection request has been initiated successfully.

Success or failure of the connection is communicated to the user through the IDeviceCallback interface. This interface must be implemented by the user to use the Camera Remote SDK.

Definition

```
CrError Connect(ICrCameraObjectInfo* pCameraObjectInfo,  IDeviceCallback*
callback, CrDeviceHandle* deviceHandle);
```

Input parameters

type	explanation
`ICrCameraObjectInfo*`	pCameraObjectInfo The camera which is going to be connected. This parameter is return by ICrEnumCameraObjectInfo::GetCameraObject().
`IDeviceCallback*`	callback The user-implemented device callback interface. App developers who use this SDK should implement the callback function interface to handle events from the camera such as connected or disconnected, property change, etc.

Input/Output parameters

type	explanation
`CrDeviceHandle*`	deviceHandle The handle of the connected camera is returned in the variable. This must be set 0 before calling Connect().

Return value

type	explanation

CrError	CrError_None on Success CrError_Init if the SDK is uninitialized CrError_Generic_Unknown If the pCameraObjectInfo is NULL, and no valid deviceNumber is supplied Other than errors above, see Status code & Error

Related API

- Disconnect
- EnumCameraObjects
- CreateCameraInfoObject
- IDeviceCallback::OnConnected

Special note (details)

This API can be used in two ways: to connect to a new device and to reconnect to an existing device.

To connect to a new device, supply a deviceHandle value of 0 and a pointer to a valid ICrCameraObjectInfo.

To reconnect to an existing device, supply the deviceHandle of that device to this API and NULL in pCameraObjectInfo.. The SDK will then reuse the existing internal device handle and attempt to connect to the specified camera device. Reconnection will not work if the specific device was previously released with the ReleaseDevice API. In this case, CrError_Generic_Unknown will be returned.

A successful connection is reported to the user through the IDeviceCallback::OnConnected interface function. An implementation of this function must be supplied to Connect by the user though the callback parameter.

The deviceHandle out-parameter returns the SDK device identifier to the user. This identifier is required to use subsequent SDK API functions to interact with the connected device.

Disconnect

Overview

This API function disconnects the indicated device.

After calling this API, the deviceHandle remains valid and can be used to reconnect to the same device.

Definition

```
CrError Disconnect(CrDeviceHandle deviceHandle);
```

Input parameters

type	explanation
`CrDeviceHandle`	deviceHandle

Output parameters

None

Return value

type	explanation
CrError	CrError_None If the deviceHandle is a valid handle. In this case, the connection to the camera will be closed. CrError_Init if the SDK is uninitialized CrError_Generic_InvalidHandle If the deviceHandle is an invalid handle Other than errors above, see Status code & Error

Related API

- Connect
- ReleaseDevice
- IDeviceCallback::OnDisconnected

Special note (details)

Stops the internal processing threads on the indicated device and disconnects from the device.

Calling this API will not invalidate the existing deviceHandle. This function simple disconnects the device. Unless ReleaseDevice is called, the device handle can be reused to connect to the same device.

The SDK signals successful disconnection by calling IDeviceCallback::OnDisconnected.

Device

ReleaseDevice

Overview

This API requests that the SDK release the resources allocated for the specified device.

Calling this API will invalidate the provided deviceHandle. Do not attempt to reuse it after calling this API.

Definition

```
CrError ReleaseDevice(CrDeviceHandle deviceHandle);
```

Input parameters

type	explanation
`CrDeviceHandle`	deviceHandle

Output parameters

None

Return value

type	explanation
CrError	CrError_None If the deviceHandle is a valid handle. CrError_Init if the SDK is uninitialized CrError_Generic_InvalidHandle If the deviceHandle is an invalid handle Other than errors above, see Status code & Error

Related API

- Connect
- Disconnect
- IDeviceCallback::OnDisconnected

Special note (details)

This function releases the resources associated with the specified device handle.

Device Property

GetDeviceProperties

Overview

This API gets device properties from the device specified by the deviceHandle.

This retrieves all of the available properties of device. This list contains information about each property's current value, list of valid values and whether or not the property value can currently be updated by the user.

Definition

```
CrError GetDeviceProperties(CrDeviceHandle deviceHandle, CrDeviceProperty**
properties, CrInt32* numOfProperties);
```

Input parameters

type	explanation
`CrDeviceHandle`	deviceHandle

Output parameters

type	explanation
CrDeviceProperty**	properties The property list pointer. Developers should pass the address of a modifiable CrDeviceProperty pointer. The value of this pointer should be initialized to nullptr. The function will make a copy of the SDK-internal CrDeviceProperty list for the indicated deviceHandle. When function returns successfully, this parameter will point to the copy of CrDeviceProperty list.
`CrInt32*`	numOfProperties A pointer to an integer which indicates the number of CrDeviceProperty objects in the property list. App developers should pass the address of a modifiable CrInt32 variable. This function will write the size of the returned list to the variable.

Return value

type	explanation

CrError	CrError_None If the properties are returned successfully CrError_Init if the SDK is uninitialized CrError_Generic_InvalidHandle If the deviceHandle is an invalid handle Other than errors above, see Status code & Error

Related API

- ReleaseDeviceProperties
- SetDeviceProperty

Special note (details)

This is a factory function. The SDK will allocate memory. Call the ReleaseDeviceProperties API to correctly release the generated list.

This API function retrieves a list of all the properties supported by the indicated device. Each returned property also provides its current value, a list of values it supports and whether or not the property is currently modifiable.

It is important to initialize the out-parameter pointer to nullptr before passing it to this function. This is required to detect whether or not a list has been created. The out-parameter properties will remain unmodified if the property list cannot be retrieved.

If the list is successfully retrieved, properties points to the list and out-parameter numOfProperties indicates the number of items in the list.

ReleaseDeviceProperties

Overview

This API function releases the CrDeviceProperty list allocated by GetDeviceProperties.

Definition

```
CrError ReleaseDeviceProperties(CrDeviceHandle deviceHandle, CrDeviceProperty*
properties);
```

Input parameters

type	explanation
`CrDeviceHandle`	deviceHandle
`CrDeviceProperty*`	properties The property list pointer pointing to the list to be released.

Output parameters

None

Return value

type	explanation
CrError	CrError_None If the property list is released successfully CrError_Init if the SDK is uninitialized CrError_Generic_InvalidHandle If the deviceHandle is an invalid handle Other than errors above, see Status code & Error

Related API

- GetDeviceProperties

Special note (details)

This function releases the CrDeviceProperty list that is associated with the specified device handle.

SetDeviceProperty

Overview

Request the SDK set a new value to the selected property for the corresponding device.

The function is asynchronous and returns to the user as soon as the SDK enqueues the requested action. After the property of the camera changed, OnPropertyChanged() call back function is called from camera, then GetDeviceProperties() will return the new property value.

Definition

```
CrError SetDeviceProperty(CrDeviceHandle deviceHandle, CrDeviceProperty*
pProperty);
```

Input parameters

type	explanation
`CrDeviceHandle`	deviceHandle
`CrDeviceProperty*`	pProperty This parameter points to the CrDeviceProperty object which contains the property that will be set to the device.

Output parameters

None

Return value

type	explanation
CrError	CrError_None If the command is sent out. CrError_Init if the SDK is uninitialized CrError_Generic_InvalidHandle If the deviceHandle is an invalid handle Other than errors above, see Status code & Error

Related API

- GetDeviceProperties

Special note (details)

Requests the SDK set the indicated pProperty on the corresponding device indicated by deviceHandle.

pProperty contains the desired property code and desired property value.

The desired value should be one of the valid values retrieved from GetDeviceProperties. The SDK will not set an unsupported value.

The return value from this function will not indicate whether or not the property was set successfully. If the property is updated successfully the SDK will call IDeviceCallback:: OnPropertyChanged(). The warning code will indicate that a property has changed.

Send Command

SendCommand

Overview

This API function sends commands for controlling the device. This allows the user to control camera functions such as the shutter release. When stop continuous shooting, use “CrCommnadId_Release” with “CrCommandParam_Up”.

The function is asynchronous and returns to the user as soon as the SDK enqueues the requested action. The effects of sending a command can be confirmed by observing the actual device for the requested change.

Definition

```
CrError SendCommand(CrDeviceHandle deviceHandle, CrInt32u commandId,
CrCommandParam commandParam);
```

Input parameters

type	explanation
`CrDeviceHandle`	deviceHandle
`CrInt32u`	commandId This parameter is one of CrCommandId defined in CrCommandData.h.
`CrCommandParam`	commandParam This parameter is one of CrCommandParam defined in CrCommandData.h.

Output parameters

None

Return value

type	explanation
CrError	CrError_None If the command is sent out. CrError_Init if the SDK is uninitialized CrError_Generic_InvalidHandle If the deviceHandle is an invalid handle Other than errors above, see Status code & Error

Related API

- SetDeviceProperty

Special note (details)

Requests the SDK send a command to the device indicated by deviceHandle.

The command to send is identified by commandId.

LiveView

GetLiveViewImage

Overview

Get the latest frame from SDK live-view image buffer.

Use the GetLiveViewImageInfo API to get information about the data size of the image before calling this API to fetch the data.

Using this data, the user can render a live preview of the camera device view finder. This data is in JPEG format.

Definition

```
CrError GetLiveViewImage(CrDeviceHandle deviceHandle,
CrImageDataBlock*imageData);
```

Input parameters

type	explanation
`CrDeviceHandle`	deviceHandle

Output parameters

type	explanation
`CrImageDataBlock*`	imageData This parameter points to an CrImageDataBlock object which is a memory buffer for storing the image data.

Return value

type	explanation
CrError	CrError_None If the live-view image data returns successfully CrError_Connect_Disconnected If the camera is not connected CrError_Init if the SDK is uninitialized CrError_Generic_InvalidHandle If the deviceHandle is an invalid handle Other than errors above, see Status code & Error

Related API

- GetLiveViewImageInfo

Special note (details)

This function retrieves one frame from the corresponding device live-view.

Before you call this function, you should call GetLiveViewImageInfo first and allocate an appropriately sized buffer for the imageData parameter.

This function does not send or receive any data from the device but merely copy the live image data from a buffer, the buffer is updated in real time by background task.

GetLiveViewImageInfo

Overview

This function returns the data size of the live-view image.

Definition

```
CrInt32u GetLiveViewImageInfo(CrDeviceHandle deviceHandle, CrImageInfo* info);
```

Input parameters

type	explanation
`CrDeviceHandle`	deviceHandle

Output parameters

type	explanation
`CrImageInfo*`	info This parameter points to a CrImageInfo object. If function returns successfully, the member bufferSize of the CrImageInfo object will be set appropriately according to the live-view image settings.

Return value

type	explanation
CrError	CrError_None If the CrImageInfo is properly set CrError_Connect_Disconnected If the camera is not connected CrError_Init if the SDK is uninitialized CrError_Generic_InvalidHandle If the deviceHandle is an invalid handle Other than errors above, see Status code & Error

Related API

- GetLiveViewImage

Special note (details)

This function is used to retrieve the size of the live-view image. Use the retrieved value to create a buffer to store the live-view image.

Call this function prior to calling GetLiveViewImage.

GetLiveViewProperties

Overview

Get live view properties from the specified device. Functionally equivalent to GetProperties for properties related to the device live-view.

The properties retrieved by this API call are closely related to the camera live-view image. These properties are not included in the list of properties retrieved by GetDeviceProperties.

Definition

```
CrError GetLiveViewProperties(CrDeviceHandle deviceHandle,
CrLiveViewProperty** properties, CrInt32* numOfProperties);
```

Input parameters

type	explanation
`CrDeviceHandle`	deviceHandle

Output parameters

type	explanation
`CrLiveViewProperty**`	properties The property list pointer. Developers should pass the address of a modifiable CrLiveViewProperty pointer. The value of this pointer should be initialized to nullptr. The function will make a copy of the SDK-internal CrLiveViewProperty list for the indicated deviceHandle. When function returns successfully, this parameter will point to the copy of CrLiveViewProperty list. Must be freed with ReleaseLiveViewProperties() after use.
`CrInt32*`	numOfProperties A pointer to an interger which indicates the number of CrLiveViewProperty objects in the property list. App developers should pass the address of a modifiable CrInt32 variable. This function will write the size of the returned list to this location.

Return value

type	explanation

CrError	CrError_None If the function returns successfully CrError_Init if the SDK is uninitialized CrError_Generic_InvalidHandle If the deviceHandle is an invalid handle Other than errors above, see Status code & Error

Related API

- ReleaseLiveViewProperties

Special note (details)

This is a factory function. The SDK will allocate memory if required.

This API function retrieves a list of all the live-view properties supported by the indicated device. Each returned property also provides its current value, a list of values it supports and whether or not the property is currently modifiable.

The out-parameter properties will remain unmodified if the property list cannot be retrieved.

If the list is successfully retrieved, properties points to the list and out-parameter numOfProperties indicates the number of items in the list.

ReleaseLiveViewProperties

Overview
This API function releases the CrLiveViewProperty list allocated by GetLiveViewProperties.

Definition
```
CrError ReleaseLiveViewProperties(CrDeviceHandle deviceHandle,
CrLiveViewProperty* properties);
```

Input parameters

type	explanation
`CrDeviceHandle`	deviceHandle
`CrLiveViewProperty*`	properties The live-view property list pointer pointing to the list to be released.

Output parameters

None

Return value

type	explanation
CrError	CrError_None If the function returns successfully CrError_Init if the SDK is uninitialized CrError_Generic_InvalidHandle If the deviceHandle is an invalid handle Other than errors above, see Status code & Error

Related API
- GetLiveViewProperties

Special note (details)
Allows the SDK to release the SDK-allocated memory for the corresponding device live-view properties list.

Supply a connected device handle.

Device Setting

GetDeviceSetting

Overview

This function returns SDK settings for the specified device.

This API can be used query to enable or disable status of live-view information for a device.

Definition

```
CrError GetDeviceSetting(CrDeviceHandle deviceHandle, CrInt32u key, CrInt32u*
value);
```

Input parameters

type	explanation
`CrDeviceHandle`	deviceHandle
`CrInt32u`	key Key for the setting to retrieve. Values can be found in the SettingKey enumeration.

Output parameters

type	explanation
`CrInt32*`	value The current value of the key in question. App developers should pass the address of a modifiable CrInt32 object. This function will write the current value of the key of interest here.

Return value

type	explanation
CrError	CrError_None If the function returns successfully CrError_Init if the SDK is uninitialized CrError_Generic_InvalidHandle If the deviceHandle is an invalid handle Other than errors above, see Status code & Error.

Related API

- SetDeviceSetting

Special note (details)

SetDeviceSetting

Overview

This API updates SDK settings for the indicated device.

This API can be used to enable or disable the live-view information for a device.

Definition

```
CrError SetDeviceSetting(CrDeviceHandle deviceHandle, CrInt32u key, CrInt32u
value);
```

Input parameters

type	explanation
`CrDeviceHandle`	deviceHandle
`CrInt32u`	key Key for the setting to update.
`CrInt32u`	value The new value for key.

Output parameters

None

Return value

type	explanation
CrError	CrError_None If the function returns successfully CrError_Init if the SDK is uninitialized CrError_Generic_InvalidHandle If the deviceHandle is an invalid handle Other than errors above, see Status code & Error

Related API

- GetDeviceSetting

Special note (details)

SetSaveInfo

Overview

This function sets the location on the PC for saving images transferred from the device.

Definition

```
CrError SetSaveInfo(CrDeviceHandle deviceHandle, CrChar* path, CrChar* prefix,
CrInt32 no);
```

Input parameters

type	explanation
`CrDeviceHandle`	deviceHandle
`CrChar*`	path The local path where images should be saved.
`CrChar*`	prefix The prefix to give saved images.
`CrInt32`	no The starting value to use when enumerating images.

Output parameters

None

Return value

type	explanation
CrError	CrError_None If the function returns successfully CrError_Init if the SDK is uninitialized CrError_Generic_InvalidHandle If the deviceHandle is an invalid handle Other than errors above, see Status code & Error

Related API

Special note (details)

The save path should be set to a location for which the application has write access.

SDK Version

GetSDKVersion

Overview

This function returns the SDK version number.

Definition

```
CrInt32u GetSDKVersion();
```

Input parameters

None

Output parameters

None

Return value

type	explanation
CrInt32u	The SDK Version is represented as a 4-byte unsigned integer constant. The first 3 bytes contain the SDK version. The last byte is reserved by the SDK for future use.

Error Codes

No Error

Related API

- GetSDKSerial

Special note (details)

The SDK version number is set at build time.
This version number will be updated if the SDK API is changed.

SDK Serial Number

GetSDKSerial

Overview

This function returns the SDK serial number.

Definition

```
CrInt32u GetSDKSerial();
```

Input parameters

None

Output parameters

None

Return value

type	explanation
CrInt32u	The SDK Serial is represented as a 4-byte unsigned integer constant. The last 2 bytes contain the SDK serial. The first 2 byte is reserved by the SDK for future use.

Error Codes

No Error

Related API

- GetSDKVersion

Special note (details)

The SDK serial number is set at build time.

Command

CrCommandId

Enumalation value describing command data type

Supported Command

Command supported in the current release.

Name	Summary
CrCommandId_Release	Set the shutter button release
CrCommandId_MovieRecord	Control Movie Rec button
CrCommandId_MediaFormat	Execute Format the Media.

Device Property

CrDeviceProperty

Class describing device properties.

Includes information about the data type, current value, and supported values. Additionally, it indicates if the property is currently modifiable.

Member Variables

Name	Type	Summary
code	CrInt32u	The data code used by this device property Defined in CrDeviceProperty.h as CrDevicePropertyCode
valueType	CrDataType	Identifies the data type used by this device property Defined in CrDefines.h as CrDataType
enableFlag	CrPropertyEnableFlag	Defines what actions can be performed on this property Defined in CrDeviceProperty.h as CrPropertyEnableFlag
currentValue	CrInt64u	The current value of this device property
valuesSize	CrInt32u	The number of items in the supported values list.
values	CrInt8u*	The supported values list of this device property. Only these values will be accepted by the SDK when using SetDeviceProperty

Member Functions

Signature	Description
Constructor	-
Destructor	-
Copy Constructor	-
bool IsGetEnabledCurrentValue()	Checks to see if property is readable
bool IsSetEnabledCurrentValue()	Checks to see if property is writable
CrInt32u GetCode()	Get the code
CrDataType GetValueType()	Get the valueType
CrPropertyEnableFlag GetPropertyEnableFlag()	Get the enableFlag
CrInt64u GetCurrentValue()	Get the currentValue
CrInt32u GetValueSize()	Get the valuesSize
CrInt8u* GetValues()	Get the values pointer
void SetValueType(CrDataType type)	Set the valueType to update

void SetCode(CrInt32u code)	Set the code to update
void SetCurrentValue(CrInt64u value)	Set the currentValue to update

Supported Properties

Properties supported in the current release.

Name	Summary
CrDeviceProperty_S1	Get/Set the shutter button half release
CrDeviceProperty_AEL	Get the AELock Indication and control AEL button
CrDeviceProperty_FEL	Get the FEL Lock Indication and control FEL button
CrDeviceProperty_AWBL	Get the AWBLock Indication and control AWBL buton
CrDeviceProperty_FNumber	Get/Set the Aperture Value （F-Number)
CrDeviceProperty_ExposureBiasCompensation	Get/Set the Exposure Bias Compensation
CrDeviceProperty_FlashCompensation	Get/Set the Flash Compensation
CrDeviceProperty_ShutterSpeed	Get/Set the Shutter Speed
CrDeviceProperty_IsoSensitivity	Get/Set the ISO Sensitivity
CrDeviceProperty_FocusArea	Get/Set the Focus Area
CrDeviceProperty_ExposureProgramMode	Get/Set the Exposure Program Mode
CrDeviceProperty_CompressionFileFormatStill	Get/Set the Compression File Format (Still)
CrDeviceProperty_FileType	Get/Set the File Format (Still)
CrDeviceProperty_JpegQuality	Get/Set the JPEG Quality
CrDeviceProperty_WhiteBalance	Get/Set the WhiteBalance
CrDeviceProperty_FocusMode	Get/Set the Focus Mode
CrDeviceProperty_MeteringMode	Get/Set the Exposure Metering Mode
CrDeviceProperty_FlashMode	Get/Set the Flash Mode
CrDeviceProperty_WirelessFlash	Get/Set the Wireless Flash Setting
CrDeviceProperty_RedEyeReduction	Get/Set the Red Eye Reduction
CrDeviceProperty_DriveMode	Get/Set the Drive Mode （Stiil Capture Mode)
CrDeviceProperty_DRO	Get/Set the Dynamic Range Optimizer
CrDeviceProperty_ImageSize	Get/Set the Image Size
CrDeviceProperty_AspectRatio	Get/Set the Aspect Ratio
CrDeviceProperty_PictureEffect	Get/Set the Picture Effect Value

CrDeviceProperty_Colortemp	Get/Set the Color Temperature
CrDeviceProperty_ColorTuningAB	Get/Set the Biaxial Fine Tuning A-B Direction
CrDeviceProperty_ColorTuningGM	Get/Set the Biaxial Fine Tuning G-M Direction
CrDeviceProperty_LiveViewDisplayEffect	Get/Set the Live View Display Effect
CrDeviceProperty_StillImageStoreDestination	Get the information of Still Image Save Destination
CrDeviceProperty_PriorityKeySettings	Get/Set the Position Key Setting
CrDeviceProperty_Focus_Magnifier_Setting	Get/Set the Focus Magnifier Setting
CrDeviceProperty_NearFar	Get/Set the Near/Far
CrDeviceProperty_AF_Area_Position	Execute set AF Area Position(x,y)
CrDeviceProperty_Zoom_Scale	Get/Set the Zoom Scale
CrDeviceProperty_Zoom_Setting	Get/Set the Zoom Setting
CrDeviceProperty_Zoom_Operation	Execute the Zoom Operation
CrDeviceProperty_Movie_File_Format	Get/Set the File Format(Movie)
CrDeviceProperty_Movie_Recording_Setting	Get/Set the Recording Setting(Movie)
CrDeviceProperty_Movie_Recording_FrameRate Setting	Get/Set the Recording Frame Rate Setting (Movie)
CrDeviceProperty_Interval_Rec_Mode	Get the Interval REC Mode
CrDeviceProperty_Still_Image_Trans_Size	Get/Set the Still Image Trans Size
CrDeviceProperty_RAW_J_PC_Save_Image	Get/Set the RAW+J PC Save Image
CrDeviceProperty_LiveView_Image_Quality	Get/Set the LiveView Quality
CrDeviceProperty_CustomWB_Capture_Standby	Get/Set the Custom WB Capture Standby
CrDeviceProperty_CustomWB_Capture_Standby_Cancel	Get/Set the Custom WB Capture Standby Cancel
CrDeviceProperty_CustomWB_Capture	Execute the Custom WB Capture
CrDeviceProperty_SnapshotInfo	Get the Shooting File Info
CrDeviceProperty_BatteryRemain	Get the Battery Remaining (%)
CrDeviceProperty_BatteryLevel	Get the Battery Level Indicator
CrDeviceProperty_RecordingState	Get the Movie Recording State
CrDeviceProperty_LiveViewStatus	LiveView Status
CrDeviceProperty_FocusIndication	Get the Focus Indication
CrDeviceProperty_MediaSLOT1_Status	Get the Media (SLOT1) Status

CrDeviceProperty_MediaSLOT1_RemainingNumber	Get the Remaining number shots of Media (SLOT1)
CrDeviceProperty_MediaSLOT1_RemainingTime	Get the Remaining shooting time of Media (SLOT1)
CrDeviceProperty_MediaSLOT1_FormatEnableStatus	Get the Media Format Enable Status(SLOT1)
CrDeviceProperty_MediaSLOT2_Status	Get the Media (SLOT2) Status
CrDeviceProperty_MediaSLOT2_RemainingNumber	Get the Remaining number shots of Media (SLOT2)
CrDeviceProperty_MediaSLOT2_RemainingTime	Get the Remaining shooting time of Media (SLOT2)
CrDeviceProperty_MediaSLOT2_FormatEnableStatus	Get the Media Format Enable Status(SLOT2)
CrDeviceProperty_Media_FormatProgressRate	Get the Media Format Progress Rate
CrDeviceProperty_Interval_Rec_Status	Get the Interval REC Status
CrDeviceProperty_CustomWB_Execution_State	Get the Custom WB Execution State
CrDeviceProperty_CustomWB_Capturable_Area	Get the Custom WB Capturable Area(x,y)
CrDeviceProperty_CustomWB_Capture_Frame_Size	Get the Custom WB Capture Frame Size(x,y)
CrDeviceProperty_CustomWB_Capture_Operation	Get the Custom WB Capture Operation Enable Status
CrDeviceProperty_Zoom_Operation_Status	Get the Zoom Operation Enable Status
CrDeviceProperty_Zoom_Bar_Information	Get the Zoom Bar Information
CrDeviceProperty_Zoom_Type_Status	Get the Zoom Type Status

Live View

CrLiveViewProperty

Class for manipulating live-view properties of a device.

Member Variables

Name	Type	Summary
code	CrInt32u	The data code used by this live-view property Defined in CrDeviceProperty.h as CrDevicePropertyCode
enableFlag	CrPropertyEnableFlag	The enableFlag used by this live-view property Defined in CrDeviceProperty.h as CrPropertyEnableFlag
valueType	CrFrameInfoType	The data type used by this live-view property Defined in CrDeviceProperty.h as CrFrameInfoType
valuesSize	CrInt32u	The valuesSize used by this live-view property
values	CrInt8u*	The values used by this live-view property

Member Functions

Signature	Description
Constructor	-
Destructor	-
Copy Constructor	-
bool IsGetEnabledCurrentValue()	Checks to see if property is readable
CrInt32u GetCode()	Get the code
CrPropertyEnableFlag GetPropertyEnableFlag()	Get the enableFlag
CrFrameInfoType GetFrameInfoType()	Get the valueType
CrInt32u GetValueSize()	Get the valuesSize
CrInt8u* GetValue()	Get the values pointer

CrFocusFrameInfo

Used to retrieve live-view frame info.

Member Variables

Name	Type	Summary
type	CrFocusFrameType	The type of focus used Defined in CrDeviceProperty.h as CrFocusFrameType
state	CrFocusFrameState	The state of the focus frame Defined in CrDeviceProperty.h as CrFocusFrameState
priority	CrInt8u	-
xNumerator	CrInt32u	x-axis value
xDenominator	CrInt32u	x-axis value
yNumerator	CrInt32u	y-axis value
yDenominator	CrInt32u	y-axis value
width	CrInt32u	Width of live-view
height	CrInt32u	Height of live-view

Member Functions

Signature	Description
Constructor	-
Destructor	-

CrMagPosInfo

Used to retrieve MagnifierPosition info.

Member Variables

Name	Type	Summary
xNumerator	CrInt32u	x-axis value
xDenominator	CrInt32u	x-axis value
yNumerator	CrInt32u	y-axis value
yDenominator	CrInt32u	y-axis value
width	CrInt32u	Width of live-view
height	CrInt32u	Height of live-view

Member Functions

Signature	Description
Constructor	-
Destructor	-

CrImageInfo

Used to retrieve live-view image info. Use this class to retrieve the size of the live-view image.

Member Variables

Name	Type	Summary
width	CrInt32u	The width of the live-view image
height	CrInt32u	The height of the live-view image
bufferSize	CrInt32u	The size of the latest live-view image in bytes

Member Functions

Signature	Description
Constructor	-
Destructor	-

CrImageDataBlock

Used for retrieving live-view image data. Allocate an object of this type to use as an output buffer.

Member Variables

Name	Type	Summary
frameNo	CrInt32u	Frame counter
size	CrInt32u	The size of the user allocated buffer, pData, in bytes
pData	CrInt8u*	Pointer to the output buffer allocated by the user

Member Functions

Signature	Description
Constructor	-
Destructor	-
CrInt32u GetFrameNo()	Get the frameNo
void SetSize(CrInt32u size)	Set the size
CrInt32u GetSize()	Get the size
void SetData(CrInt8u* data)	Set the pData
CrInt32u GetImageSize()	Get the jpeg data size
CrInt8u* GetImageData()	Get the jpeg data

Callback Interface

IDeviceCallback

The callback interface of the SDK. This interface is used by the Camera Remote SDK to communicate the result of various asynchronous events to the user.

The user must implement a class deriving from this interface to use the SDK. This derived class should be passed to the Connect API to establish the callback communication channel with the SDK.

Pure Virtual Functions

Signature	Description
virtual void OnConnected(DeviceConnectionVersion version)	Called by the SDK when a device is successfully connected
virtual void OnDisconnected(CrInt32u error)	Called by the SDK when a device disconnects. The error code may indicate a reason
virtual void OnPropertyChanged()	Called by the SDK when a device property changes
virtual void OnCompleteDownload()	Called by the SDK when a photo has completely been transferred to the user device
virtual void OnWarning(CrInt32u warning)	Called when the SDK detects a warning. The warning code is passed back to the application as a parameter
virtual void OnError(CrInt32u error)	Called when the SDK detects an error. The error code is passed back to the application as a parameter

ICrCameraObjectInfo

Your application can access to the specified camera information that is enumerated by EnumCameraObjectInfo() using this interface.

The information retrieved from this interface is useful for displaying various information about the corresponding device to the end user of an application utilising the Camera Remote SDK. The information provided by this class is also required when establishing a new connection to a camera device. It should be provided when calling the Connect API.

The user should never manually free these objects by calling free or delete. Instead, the user should call ICrCameraObjectInfo::Release. This passes responsibility for releasing the allocated memory to the SDK, where it can be properly released.

Pure Virtual Functions

Signature	Description
virtual void Release()	Calls the SDK to destroy the allocated object
virtual CrChar* GetName() const	Gets the friendly device name as a null-terminated character string (Friendly device name is not available through SDK, currently.)
virtual CrInt32u GetNameSize() const	Gets the size of the name string
virtual CrChar* GetModel() const	Gets the device model name as a null-terminated character string
virtual CrInt32u GetModelSize() const	Gets the size of the model string
virtual CrInt16 GetUsbPid(CrInt32u error) const	Gets the product id of a USB device
virtual CrInt8u* GetId() const	Gets the pointer to the device id data buffer
virtual CrInt32u GetIdSize() const	Gets the id data size
virtual CrInt32u GetIdType() const	Gets the id data type (binary or string data)
virtual CrInt32u GetConnectionStatus() const	Gets the current connection status of the device
virtual CrChar* GetConnectionTypeName() const	Gets the connection type string
virtual CrChar* GetAdaptorName() const	Gets the adaptor name string

ICrEnumCameraObjectInfo

The virtual interface for interacting with enumerated device info list created by the SDK.

This is the enumerator object interface to access the list of connectable cameras. Your application can get the access interface to the each camera using GetCameraObjectInfo().

A "connectable" device fulfils three requirements. One, the device itself supports PC Remote Control features. Two, the device model is supported by the current Camera Remote SDK release. Three, the connection method used by the device is supported by the current Camera Remote SDK. All three requirements must be fulfilled for the device information to be populated in the list.

All ICrEnumCameraObjectInfo interface objects are allocated internally by the SDK before having their address passed back to the user. The user should never manually free these objects by calling free or delete. Instead, the user should call ICrEnumCameraObjectInfo::Release. This passes responsibility for releasing the allocated memory to the SDK, where it can be properly released.

Pure Virtual Functions

Signature	Description
virtual void Release()	Calls the SDK to destroy the allocated device info list
virtual CrInt32u GetCount() const	Returns the number of device info objects in the allocated list
virtual const ICrCameraObjectInfo* GetCameraObjectInfo(CrInt32u index) const	Get a pointer to the ICrCameraObjectInfo at the index specified

Status code & Error

Major status codes are below. The "error" member is defined as [error_code, error_message].
The error_message may vary depending on the camera models.

Error Category

Value	Name	Summary
0x0000	CrError_None	No error
0x8000	CrError_Generic	Uncategorized errors
0x8100	CrError_File	File errors
0x8200	CrError_Connect	Communication errors
0x8300	CrError_Memory	Memory errors
0x8400	CrError_Api	API errors
0x8500	CrError_Init	Initialization errors
0x8600	CrError_Polling	Polling errors
0x8700	CrError_Adaptor	Adapter errors
0x8800	CrError_Device	Device errors
0xc000	CrError_Application	Application-specific errors

CrError_None

CrError_Generic

Value	Name	Summary
0x8000	CrError_Generic_Unknown	Uncategorized errors
0x8001	CrError_Generic_Notimpl	Not implemented
	CrError_Generic_Abort	Processing was aborted
	CrError_Generic_NotSupported	Not supported
	CrError_Generic_SeriousErrorNotSupported	Not supported
	CrError_Generic_InvalidHandle	Not valid handle
	CrError_Generic_InvalidParameter	Invalid parameter

CrError_File

Value	Name	Summary
0x8100	CrError_File_Unknown	Unknown file errors
	CrError_File_IllegalOperation	Illegal operation (e.g., loading without opening)
	CrError_File_IllegalParameter	Illegal parameter
	CrError_File_EOF	EOF
	CrError_File_OutOfRange	Operation, such as seek, is out of range
	CrError_File_NotFound	File not found
	CrError_File_DirNotFound	Directory not found
	CrError_File_AlreadyOpened	Already opened
	CrError_File_PermissionDenied	No access permission
	CrError_File_StorageFull	Storage is full
	CrError_File_AlreadyExists	Already exists
	CrError_File_TooManyOpenedFiles	Too many open files
	CrError_File_ReadOnly	Read-Only file
	CrError_File_CantOpen	Cannot open
	CrError_File_CantClose	Cannot close
	CrError_File_CantDelete	Cannot delete
	CrError_File_CantRead	Cannot read
	CrError_File_CantWrite	Cannot write
	CrError_File_CantCreateDir	Cannot create a directory
	CrError_File_OperationAbortedByUser	Processing was aborted by user

	CrError_File_UnsupportedOperation	API not supported for the platform was called
	CrError_File_NotYetCompleted	Operation is not completed
	CrError_File_Invalid	The file is no longer valid because the volume for the file was altered
	CrError_File_StorageNotExist	The specified network resource or device is no longer available
	CrError_File_SharingViolation	Sharing violation
	CrError_File_Rotation	Invalid file orientation
	CrError_File_SameNameFull	Too many same-name files

CrError_Connect

Value	Name	Summary
0x8200	CrError_Connect_Unknown	Other errors classified as connection except below
	CrError_Connect_Connect	A connection request failed through the USB
	CrError_Connect_Release	Release failed
	CrError_Connect_GetProperty	Getting property failed
	CrError_Connect_SendCommand	Sending command failed
	CrError_Connect_HandlePlugin	Illegal handle plug-in
	CrError_Connect_Disconnected	A connection disconnected
	CrError_Connect_TimeOut	A connection operation timed out
	CrError_Reconnect_TimeOut	Reconnection operations timed out.
	CrError_Connect_FailRejected	Connection rejected and failed
	CrError_Connect_FailBusy	Connection failed due to processing in progress
	CrError_Connect_FailUnspecified	Unspecified connection failure
	CrError_Connect_Cancel	Connection canceled

CrError_Memory

Value	Name	Summary
0x8300	CrError_Memory_Unknown	Unknown memory error
	CrError_Memory_OutOfMemory	Cannot allocate memory
	CrError_Memory_InvalidPointer	Invalid pointer
	CrError_Memory_Insufficient	Allocate memory infufficient

CrError_Api

Value	Name	Summary
0x8400	CrError_Api_Unknown	Unknown API error
	CrError_Api_Insufficient	Incorrect parameter
	CrError_Api_InvalidCalled	Invalid API call

CrError_Init

Value	Name	Summary
0x8500	CrError_Init_Unknown	Unknown init error

CrError_Polling

Value	Name	Summary
0x8600	CrError_Polling_Unknown	Unknown polling error
	CrError_Polling_InvalidVal_Intervals	Invalid polling interval setting value

CrError_Adaptor

Value	Name	Summary
0x8700	CrError_Adaptor_Unknown	Unknown adapter error
	CrError_Adaptor_InvaildProperty	A property that doesn't exist was used
	CrError_Adaptor_GetInfo	Getting information failed
	CrError_Adaptor_Create	Creation failed
	CrError_Adaptor_SendCommand	Sending command failed
	CrError_Adaptor_HandlePlugin	Illegal handle plug-in
	CrError_Adaptor_CreateDevice	Device creation failed
	CrError_Adaptor_EnumDecvice	Enumeration of device information failed
	CrError_Adaptor_Reset	Reset failed
	CrError_Adaptor_Read	Read failed
	CrError_Adaptor_Phase	Parse failed
	CrError_Adaptor_DataToWiaItem	Failed to set data as WIA item
	CrError_Adaptor_DeviceBusy	The setting side is busy
	CrError_Adaptor_Escape	Escape failed

CrError_Device

Value	Name	Summary
0x8800	CrError_Device_Unknown	Unknown device error

CrWarning

Value	Name	Summary
0x00020000	CrWarning_Unknown	Warning: unknown warning
0x00020001	CrWarning_Connect_Reconnected	Warning: reconnected
0x00020002	CrWarning_Connect_Reconnecting	Warning: reconnecting
0x00020003	CrWarning_File_StorageFull	Warning: storage is almost full
0x00020004	CrWarning_SetFileName_Failed	Warning: file name setting error
0x00020005	CrWarning_GetImage_Failed	Warning: error in getting image
0x00020007	CrWarning_NetworkErrorOccurred	Warning: network error occurred
0x00020008	CrWarning_NetworkErrorRecovered	Warning: recovered from network error
0x00020009	CrWarning_Format_Failed	Warning: formatting failed
0x0002000a	CrWarning_Format_Invalid	Warning: invalid formatting
0x0002000b	CrWarning_Format_Complete	Warning: formatting complete
0x00020011	CrWarning_Exposure_Started	Warning: exposure start
0x00020017	CrWarning_Frame_NotUpdated	Warning: live view frame not update

CrNotify

Value	Name	Summary
0x00020018	CrNotify_All_Download_Complete	Notification: download completed

Parameter description

CrCommandId_Release

Release the shutter to shoot

Parameter Code	Explanation
CrCommandParam_Up	Up the shutter button
CrCommandParam_Down	Down the shutter button After executing “Down”, send “Up” to cancel the Down status.

CrCommandId_MovieRecord

Control Movie Rec button

Parameter Code	Explanation
CrCommandParam_Down	Down the movie button Specify “Down” when you start or stop movie recording

CrCommandId_MediaFormat

Formatting the media

Parameter Code	Explanation
CrCommandParam_Up	Specify when initializing the media in SLOT1 Ex. “CrCommandId_ MediaFormat” with “CrCommandParam_Up”
CrCommandParam_Down	Specify when initializing the media in SLOT2 Ex. “CrCommandId_ MediaFormat” with “CrCommandParam_Down”

CrDeviceProperty_S1

Get/Set the Shutter button half release

Parameter Code	Explanation
CrLockIndicator_Unlocked	Unlock
CrLockIndicator_Locked	Lock

CrDeviceProperty_AEL

Get the AELock Indication and control AEL button

Parameter Code	Explanation
CrLockIndicator_Unlocked	Unlock
CrLockIndicator_Locked	Lock

CrDeviceProperty_FEL

Get the FEL Lock Indication and control FEL button

Parameter Code	Explanation
CrLockIndicator_Unlocked	Unlock
CrLockIndicator_Locked	Lock

CrDeviceProperty_AWBL

Get the AWBLock Indication and control AWBL buton

Parameter Code	Explanation
CrLockIndicator_Unlocked	Unlock
CrLockIndicator_Locked	Lock

CrDeviceProperty_FNumber

Get/Set the Aperture Value（F-Number)

Value	Explanation
CrFnumber_Nothing	Nothing to display
CrFnumber_Unknown	Display "--"
Other than above values	The value is obtained by multiplying a real FNumber value by 100. e.g.) 0x0190 =400 (means F-4) 0x03B6 = 950 (means F-9.5)

CrDeviceProperty_ExposureBiasCompensation

Get/Set the Exposure Bias Compensation

Value	Explanation
-	The value is obtained by multiplying a real Exposure Bias Compensation value by 1000. e.g.) 0xEC78 = -5000 (means -5.0Ev) 0x0000 = 0 (means 0.0Ev) 0x1388 = 5000 (means 5.0Ev)

CrDeviceProperty_FlashCompensation

Get/Set the Flash Compensation

Value	Explanation
-	The value is obtained by multiplying a real Flash Compensation value by 1000. e.g.) 0xEC78 = -5000 (means -5.0Ev) 0x0000 = 0 (means 0.0Ev) 0x1388 = 5000 (means 5.0Ev)

CrDeviceProperty_ShutterSpeed

Get/Set the Shutter Speed

Value	Explanation
CrShutterSpeed_Bulb	BULB
CrShutterSpeed_Nothing	nothing to display
Other than above values	The real value of shutter speed (Upper two bytes: numerator, Lower two bytes: denominator) In the case of the shutter speed is displayed as "Real Number" on the camera, the denominator is fixed 0x000A. e.g.) 0x000F000A: 0x000F (means 15) / 0x0000A (means 10) = 1.5" In the case of the shutter speed is displayed as "Fraction Number" on the camera, the numerator is fixed 0x0001. e.g.) 0x000103E8: 0x0001 (means 1) / 0x03E8 (means 1000) = 1/1000

CrDeviceProperty_IsoSensitivity

Get/Set the ISO Sensitivity

Value	Explanation
-	value : bit 28-31 ISO mode , bit 0-27 ISO value. Real ISO value : when bits 0-27 are other than CrISO_AUTO(0xFFFFFF). e.g.) 0x00000140 = 320

CrDeviceProperty_FosucArea

Get/Set the Focus Area

Parameter Code	Explanation
CrFocusArea_Wide	Wide
CrFocusArea_Zone	Zone
CrFocusArea_Center	Center
CrFocusArea_Flexible_Spot_S	Flexible spot S
CrFocusArea_Flexible_Spot_M	Flexible spot M
CrFocusArea_Flexible_Spot_L	Flexible spot L
CrFocusArea_Expand_Flexible_Spot	Expand flexible spot
CrFocusArea_Flexible_Spot	Flexible spot

CrDeviceProperty_ExposureProgramMode

Get/Set the Exposure Program Mode

Parameter Code	Explanation
CrExposure_M_Manual	Manual(M)
CrExposure_P_Auto	Automatic(P)
CrExposure_A_AperturePriority	Aperture Priority(A)
CrExposure_S_ShutterSpeedPriority	Shutter Priority(S)
CrExposure_Program_Creative	Program Creative(greater depth of field)
CrExposure_Program_Action	Program Action(faster shutter speed)
CrExposure_Portrait	Portrait
CrExposure_Auto	Auto
CrExposure_Auto_Plus	Auto+
CrExposure_P_A	P_A
CrExposure_P_S	P_S
CrExposure_Sprots_Action	Sports Action
CrExposure_Sunset	Sunset
CrExposure_Night	Night Scene
CrExposure_Landscape	Landscape
CrExposure_Macro	Macro
CrExposure_HandheldTwilight	Hand-held Twilight
CrExposure_NightPortrait	Night Portrait
CrExposure_AntiMotionBlur	Anti Motion Blur
CrExposure_Pet	Pet
CrExposure_Gourmet	Gourmet

CrExposure_Fireworks	Fireworks
CrExposure_HighSensitivity	High Sensitivity
CrExposure_MemoryRecall	MemoryRecall(MR)
CrExposure_ContinuousPriority_AE_8pics	Tele-Zoom Continuous Priority AE 8pics
CrExposure_ContinuousPriority_AE_10pics	Tele-Zoom Continuous Priority AE 10pics
CrExposure_ContinuousPriority_AE_12pics	Continuous Priority AE12pics
CrExposure_3D_SweepPanorama	3D Sweep Panorama Shooting
CrExposure_SweepPanorama	Sweep Panorama Shooting
CrExposure_Movie_P	Movie Recording(P)
CrExposure_Movie_A	Movie Recording(A)
CrExposure_Movie_S	Movie Recording(S)
CrExposure_Movie_M	Movie Recording(M)
CrExposure_Movie_Auto	Movie Recording(AUTO)
CrExposure_Movie_SQMotion_P	Movie Recording(Slow&Quick Motion(P))
CrExposure_Movie_SQMotion_A	Movie Recording(Slow&Quick Motion(A))
CrExposure_Movie_SQMotion_S	Movie Recording(Slow&Quick Motion(S))
CrExposure_Movie_SQMotion_M	Movie Recording(Slow&Quick Motion(M))
CrExposure_Flash_Off	Flash Off
CrExposure_PictureEffect	PictureEffect
CrExposure_HiFrameRate_P	High Frame Rate(P)
CrExposure_HiFrameRate_A	High Frame Rate(A)
CrExposure_HiFrameRate_S	High Frame Rate(S)
CrExposure_HiFrameRate_M	High Frame Rate(M)
CrExposure_SQMotion_P	S&Q Motion(P)
CrExposure_SQMotion_A	S&Q Motion(A)
CrExposure_SQMotion_S	S&Q Motion(S)
CrExposure_SQMotion_M	S&Q Motion(M)
CrExposure_MOVIE	MOVIE
CrExposure_STILL	STILL

CrDeviceProperty_CompressFileFormatStill

Get/Set the Compression File Format(Still)

Depends on this setting, available settings vary at CrDeviceProperty_FileType.

Parameter Code	Explanation
CrCompressionFileFormat_JPEG	JPEG
CrCompressionFileFormat_HEIF_422	HEIF (4:2:2)
CrCompressionFileFormat_HEIF_420	HEIF (4:2:0)

CrDeviceProperty_FileType

Get/Set the File Format(Still)

Before setting this, check if CrDeviceProperty_CompressFileFormatStill is set properly.

Parameter Code	Explanation
CrFileType_RawJpeg	RAW+JPEG
CrFileType_Jpeg	JPEG
CrFileType_Raw	RAW
CrFileType_RawHeif	RAW+HEIF
CrFileType_Heif	HEIF

CrDeviceProperty_JpegQuality

Get/Set the JPEG Quality

Parameter Code	Explanation
CrJpegQuality_Standard	Standard
CrJpegQuality_Fine	Fine
CrJpegQuality_ExFine	Extra fine

CrDeviceProperty_WhiteBalance

Get/Set the WhiteBalance

Parameter Code	Explanation
CrWhiteBalance_AWB	AWB
CrWhiteBalance_Underwater_Auto	Underwater Auto
CrWhiteBalance_Daylight	Daylight
CrWhiteBalance_Shadow	Shade
CrWhiteBalance_Cloudy	Cloudy
CrWhiteBalance_Tungsten	Tungsten (Incandescent)
CrWhiteBalance_Fluorescent	Fluorescent
CrWhiteBalance_Fluorescent_WarmWhite	Fluor::Warm White(-1)
CrWhiteBalance_Fluorescent_CoolWhite	Fluor::Cool White(0)
CrWhiteBalance_Fluorescent_DayWhite	Fluor::Day White(+1)
CrWhiteBalance_Fluorescent_Daylight	Fluor::Daylight White(+2)
CrWhiteBalance_Flush	Flush
CrWhiteBalance_ColorTemp	C.Temp.
CrWhiteBalance_Custom_1	Custom1
CrWhiteBalance_Custom_2	Custom2
CrWhiteBalance_Custom_3	Custom3
CrWhiteBalance_Custom	Custom

CrDeviceProperty_FocusMode

Get/Set the Focus Mode

Parameter Code	Explanation
CrFocus_MF	Manual(MF)
CrFocus_AF_S	Automatic(AF_S)
CrFocus_AF_C	Continuous AF(AF_C)
CrFocus_AF_A	Auto(AF_A)
CrFocus_AF_D	(AF-D)
CrFocus_DMF	Direct Manual Focus(DMF)
CrFocus_PF	Preset Focus(PF)

CrDeviceProperty_MeteringMode

Get/Set the Exposure Metering Mode

Parameter Code	Explanation
CrMetering_Average	Average
CrMetering_CenterWeightedAverage	Center-weighted-average
CrMetering_MultiSpot	Multi-spot
CrMetering_CenterSpot	Center-spot
CrMetering_Multi	Multi
CrMetering_CenterWeighted	Center-weighted
CrMetering_EntireScreenAverage	Entire Screen Avg.
CrMetering_Spot_Standard	Spot：Standard
CrMetering_Spot_Large	Spot：Large
CrMetering_HighLightWeighted	Highlight

CrDeviceProperty_FlashMode

Get/Set the Flash Mode

Parameter Code	Explanation
CrFlash_Auto	Auto flash
CrFlash_Off	Flash off
CrFlash_Fill	Fill flash
CrFlash_ExternalSync	External Sync
CrFlash_SlowSync	Slow Sync
CrFlash_RearSync	Rear Sync

CrDeviceProperty_WirelessFlash

Get/Set the Wireless Flash Setting

Parameter Code	Explanation
CrWirelessFlash_Off	Off
CrWirelessFlash_On	On

CrDeviceProperty_RedEyeReduction

Get/Set the Red Eye Reduction

Parameter Code	Explanation
CrRedEye_Off	Off
CrRedEye_On	On

CrDeviceProperty_DriveMode

Get/Set the Drive Mode（Stiil Capture Mode)

Parameter Code	Explanation
CrDrive_Single	Normal
CrDrive_Continuous_Hi	Continuous Shot hi
CrDrive_Continuous_Hi_Plus	Cont. Shooting Hi+
CrDrive_Continuous_Hi_Live	Cont. Shooting Hi-Live
CrDrive_Continuous_Lo	Continuous Shot lo
CrDrive_Continuous	Continuous Shot
CrDrive_Continuous_SpeedPriority	Continuous Shot Speed Priority
CrDrive_Continuous_Mid	Continuous Shot mid
CrDrive_Continuous_Mid_Live	Cont. Shooting Mid-Live
CrDrive_Continuous_Lo_Live	Cont. Shooting Lo-Live
CrDrive_Timelapse	Timelapse
CrDrive_Timer_5s	Self Timer 5sec
CrDrive_Timer_10s	Self Timer 10sec
CrDrive_Timer_2s	Self Timer 2sec
CrDrive_Continuous_Bracket_03Ev_3pics	Continuous Bracket 0.3EV 3pics
CrDrive_Continuous_Bracket_03Ev_5pics	Continuous Bracket 0.3EV 5pics
CrDrive_Continuous_Bracket_03Ev_9pics	Continuous Bracket 0.3EV 9pics
CrDrive_Continuous_Bracket_05Ev_3pics	Continuous Bracket 0.5EV 3pics
CrDrive_Continuous_Bracket_05Ev_5pics	Continuous Bracket 0.5EV 5pics
CrDrive_Continuous_Bracket_05Ev_9pics	Continuous Bracket 0.5EV 9pics
CrDrive_Continuous_Bracket_07Ev_3pics	Continuous Bracket 0.7EV 3pics
CrDrive_Continuous_Bracket_07Ev_5pics	Continuous Bracket 0.7EV 5pics
CrDrive_Continuous_Bracket_07Ev_9pics	Continuous Bracket 0.7EV 9pics

CrDrive_Continuous_Bracket_10Ev_3pics	Continuous Bracket 1.0EV 3pics
CrDrive_Continuous_Bracket_10Ev_5pics	Continuous Bracket 1.0EV 5pics
CrDrive_Continuous_Bracket_10Ev_9pics	Continuous Bracket 1.0EV 9pics
CrDrive_Continuous_Bracket_20Ev_3pics	Continuous Bracket 2.0EV 3pics
CrDrive_Continuous_Bracket_20Ev_5pics	Continuous Bracket 2.0EV 5pics
CrDrive_Continuous_Bracket_30Ev_3pics	Continuous Bracket 3.0EV 3pics
CrDrive_Continuous_Bracket_30Ev_5pics	Continuous Bracket 3.0EV 5pics
CrDrive_Single_Bracket_03Ev_3pics	Single Bracket 0.3EV 3pics
CrDrive_Single_Bracket_03Ev_5pics	Single Bracket 0.3EV 5pics
CrDrive_Single_Bracket_03Ev_9pics	Single Bracket 0.3EV 9pics
CrDrive_Single_Bracket_05Ev_3pics	Single Bracket 0.5EV 3pics
CrDrive_Single_Bracket_05Ev_5pics	Single Bracket 0.5EV 5pics
CrDrive_Single_Bracket_05Ev_9pics	Single Bracket 0.5EV 9pics
CrDrive_Single_Bracket_07Ev_3pics	Single Bracket 0.7EV 3pics
CrDrive_Single_Bracket_07Ev_5pics	Single Bracket 0.7EV 5pics
CrDrive_Single_Bracket_07Ev_9pics	Single Bracket 0.7EV 9pics
CrDrive_Single_Bracket_10Ev_3pics	Single Bracket 1.0EV 3pics
CrDrive_Single_Bracket_10Ev_5pics	Single Bracket 1.0EV 5pics
CrDrive_Single_Bracket_10Ev_9pics	Single Bracket 1.0EV 9pics
CrDrive_Single_Bracket_20Ev_3pics	Single Bracket 2.0EV 3pics
CrDrive_Single_Bracket_20Ev_5pics	Single Bracket 2.0EV 5pics
CrDrive_Single_Bracket_30Ev_3pics	Single Bracket 3.0EV 3pics
CrDrive_Single_Bracket_30Ev_5pics	Single Bracket 3.0EV 5pics
CrDrive_WB_Bracket_Lo	WhiteBalance Bracket Lo
CrDrive_WB_Bracket_Hi	WhiteBalance Bracket Hi
CrDrive_DRO_Bracket_Lo	DRO Bracket Lo
CrDrive_DRO_Bracket_Hi	DRO Bracket Hi
CrDrive_LPF_Bracket	LPF Bracket
CrDrive_RemoteCommander	Remote Commander
CrDrive_MirrorUp	Mirror Up
CrDrive_SelfPortrait_1	Selt Portrait 1 Persion
CrDrive_SelfPortrait_2	Self Portrait 2people
CrDrive_Continuous_Timer_3pics	Continuous Self Timer 3pics

CrDrive_Continuous_Timer_5pics	Continuous Self Timer 5pics
CrDrive_Continuous_Timer_5s_3pics	Continuous Self Timer 3pics 5sec
CrDrive_Continuous_Timer_5s_5pics	Continuous Self Timer 5pics 5sec
CrDrive_Continuous_Timer_2s_3pics	Continuous Self Timer 3pics 2sec
CrDrive_Continuous_Timer_2s_5pics	Continuous Self Timer 5pics 2sec
CrDrive_SingleBurstShooting_lo	Spot Burst Shooting Lo
CrDrive_SingleBurstShooting_mid	Spot Burst Shooting Mid
CrDrive_SingleBurstShooting_hi	Spot Burst Shooting Hi

CrDeviceProperty_DRO

Get/Set the Dynamic Range Optimizer

Parameter Code	Explanation
CrDRangeOptimizer_Off	DRO OFF
CrDRangeOptimizer_On	DRO
CrDRangeOptimizer_Plus	DRO+
CrDRangeOptimizer_Plus_Manual_1	DRO + Manual1
CrDRangeOptimizer_Plus_Manual_2	DRO + Manual2
CrDRangeOptimizer_Plus_Manual_3	DRO + Manual3
CrDRangeOptimizer_Plus_Manual_4	DRO + Manual4
CrDRangeOptimizer_Plus_Manual_5	DRO + Manual5
CrDRangeOptimizer_Auto	DRO AUTO
CrDRangeOptimizer_HDR_Auto	HDR AUTO
CrDRangeOptimizer_HDR_10Ev	HDR 1.0Ev
CrDRangeOptimizer_HDR_20Ev	HDR 2.0Ev
CrDRangeOptimizer_HDR_30Ev	HDR 3.0Ev
CrDRangeOptimizer_HDR_40Ev	HDR 4.0Ev
CrDRangeOptimizer_HDR_50Ev	HDR 5.0Ev
CrDRangeOptimizer_HDR_60Ev	HDR 6.0Ev

CrDeviceProperty_ImageSize

Get/Set the Image Size

Parameter Code	Explanation
CrImageSize_L	L
CrImageSize_M	M
CrImageSize_S	S
CrImageSize_VGA	VGA

CrDeviceProperty_AspectRatio

Get/Set the Aspect Ratio

Parameter Code	Explanation
CrAspectRatio_3_2	3:2
CrAspectRatio_16_9	16:9
CrAspectRatio_4_3	4:3
CrAspectRatio_1_1	1:1

CrDeviceProperty_PictureEffect

Get/Set the Picture Effect Value

Parameter Code	Explanation
CrPictureEffect_Off	OFF
CrPictureEffect_ToyCameraNormal	Toy Camera Normal
CrPictureEffect_ToyCameraCool	Toy Camera Cool
CrPictureEffect_ToyCameraWarm	Toy Camera Warm
CrPictureEffect_ToyCameraGreen	Toy Camera Green
CrPictureEffect_ToyCameraMagenta	Toy Camera Magenta
CrPictureEffect_Pop	Pop Color
CrPictureEffect_PosterizationBW	Posterization B/W
CrPictureEffect_PosterizationColor	Posterization Color
CrPictureEffect_Retro	Retro Photo
CrPictureEffect_SoftHighkey	Soft High-key
CrPictureEffect_PartColorRed	Partial Color Red
CrPictureEffect_PartColorGreen	Partial Color Green

CrPictureEffect_PartColorBlue	Partial Color Blue
CrPictureEffect_PartColorYellow	Partial Color Yellow
CrPictureEffect_HighContrastMonochrome	High Contrast Mono
CrPictureEffect_SoftFocusLow	Soft Focus Low
CrPictureEffect_SoftFocusMid	Soft Focus Mid
CrPictureEffect_SoftFocusHigh	Soft Focus High
CrPictureEffect_HDRPaintingLow	HDR Painting Low
CrPictureEffect_HDRPaintingMid	HDR Painting Mid
CrPictureEffect_HDRPaintingHigh	HDR Painting High
CrPictureEffect_RichToneMonochrome	Rich-tone Mono
CrPictureEffect_MiniatureAuto	Miniature Auto
CrPictureEffect_MiniatureTop	Miniature Top
CrPictureEffect_MiniatureMidHorizontal	Miniature Middle(Horizontal)
CrPictureEffect_MiniatureBottom	Miniature Bottom
CrPictureEffect_MiniatureLeft	Miniature Left
CrPictureEffect_MiniatureMidVertical	Miniature Middle(Vertical)
CrPictureEffect_MiniatureRight	Miniature Right
CrPictureEffect_MiniatureWaterColor	Miniature Wator Color
CrPictureEffect_MiniatureIllustrationLow	Miniature Illustration Low
CrPictureEffect_MiniatureIllustrationMid	Miniature Illustration Mid
CrPictureEffect_MiniatureIllustrationHigh	Miniature Illustration High

CrDeviceProperty_Colortemp

Get/Set the Color Temperature

Value	Explanation	
0x09C4(2500K)	min	Values are following. - 0x0000 means less than 2500K. - 0xFFFF means greater than 9900K.
0x26AC(9900K)	max	
0x0064(100K)	step	

CrDeviceProperty_ColorTuningAB

Get/Set the Biaxial Fine Tuning A-B Direction

Value	Explanation	
0x9C(B9_00)	min	AB value sent to PC App from camera corresponds to one of the following patterns. AB number is BY or AY, where Y is decimal from 0.00 to 9.00 and increments by 0.25. Ex.) B9.00(0x9C), B8.75(0x9D), ..., A8.75(0xE3), A9.00(0xE4). Note: There may be parameter scope differences due to model differences.
0xE4(A9_00)	max	
0x01(0.25)	step	

CrDeviceProperty_ColorTuningGM

Get/Set the Biaxial Fine Tuning G-M Direction

Value	Explanation	
0x9C(M9_00)	min	GM value sent to PC App from camera corresponds to one of the following patterns. GM number is MX or GX, where X is decimal from 0.00 to 9.00 and increments by 0.25. Ex.) M9.00(0x9C), M8.75(0x9D), ..., G8.75(0xE3), G9.00(0xE4). Note: There may be parameter differences due to model differences.
0xE4(G9_00)	max	
0x01(0.25)	step	

CrDeviceProperty_LiveViewDisplayEffect

Get/Set the Live View Display Effect

Parameter Code	Explanation
CrLiveViewDisplayEffect_Unknown	Unknown
CrLiveViewDisplayEffect_ON	Effect ON
CrLiveViewDisplayEffect _OFF	Effect OFF

CrDeviceProperty_StillImageStoreDestination

Get the information of Still Image Save Destination

Parameter Code	Explanation
CrStillImageStoreDestination_HostPC	Host Device (Ex. PC)
CrStillImageStoreDestination_MemoryCard	Camera(Memory Card)
CrStillImageStoreDestination_HostPCAndMemoryCard	Host Device & Camera(Memory Card)

CrDeviceProperty_PriorityKeySettings

Get/Set the Position Key Setting

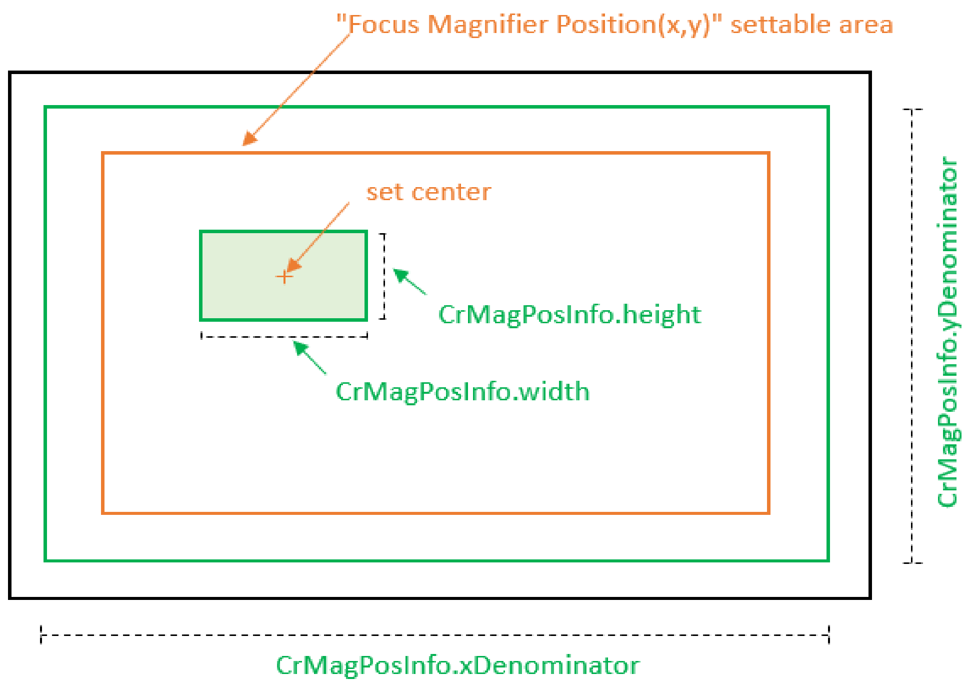
Parameter Code	Explanation
CrPriorityKey_CameraPosition	Camera position priority
CrPriorityKey_PCRemote	PC Remote setting priority

CrDeviceProperty_Focus_Magnifier_Setting

Get/Set the Focus Magnifier Setting

Value	Explanation
0x0000000000000000 ～ 0xFFFFFFFFFFFFFFFF	The upper 4 bytes are the Focus Magnifier Ratio, and the lower 4 bytes are the Focus Magnifier Position(x,y). Caution : The range of focus magnifier ratio and focus magnifier position varies depending on the model and aspect ratio. [Upper 4bytes] Regarding Focus Magnifier Ratio : Select the focus magnifier ratio to be set from the focus magnifier ratio obtained by GetValues() function. Ex.) Result obtained by GetValues() function. If the camera supports OFF, x1.0, x4.0 and x8.0 as focus magnifier ratio, Result is the following. Enum value[0] = 0x00000000FFFFFFFF (means OFF) Enum value[1] = 0x0000000AFFFFFFFF (means x1.0) Enum value[2] = 0x00000028FFFFFFFF (means x4.0) Enum value[3] = 0x00000050FFFFFFFF (means x8.0) [Lower 4bytes] Regarding Focus Magnifier Position (x,y) : The upper 2 bytes are the x coordinate and the lower 2 bytes are the y coordinate. If focus magnifier position (x) is 150 and (y) is 100, set 0x00960064. 0x0096 = 0d150, 0x0064 = 0d100. The range of X is 0～639 (0x027F), and the range of Y is 0～479 (0x01DF). Frame size is acquired by CrMagPosInfo. CrMagPosInfo is in LiveViewProperty. Since this position specifies the center of the frame, the position range is more inside by half the frame size than CrMagPosInfo.xDenominator/yDenominator. Note: See Tips/Trouble shooting for a detailed implementation example. Focus Magnifier Setting

Fig. Relationship between CrMagPosInfo and settable area


"Focus Magnifier Position(x,y)" settable area
set center
CrMagPosInfo.height
CrMagPosInfo.width
CrMagPosInfo.yDenominator
CrMagPosInfo.xDenominator

CrDeviceProperty_NearFar

Get the Focus Near/Far Enable Status

Parameter Code	Explanation
CrNearFar_Disable	Disable
CrNearFar_Enable	Enable

Set the Focus Near/Far

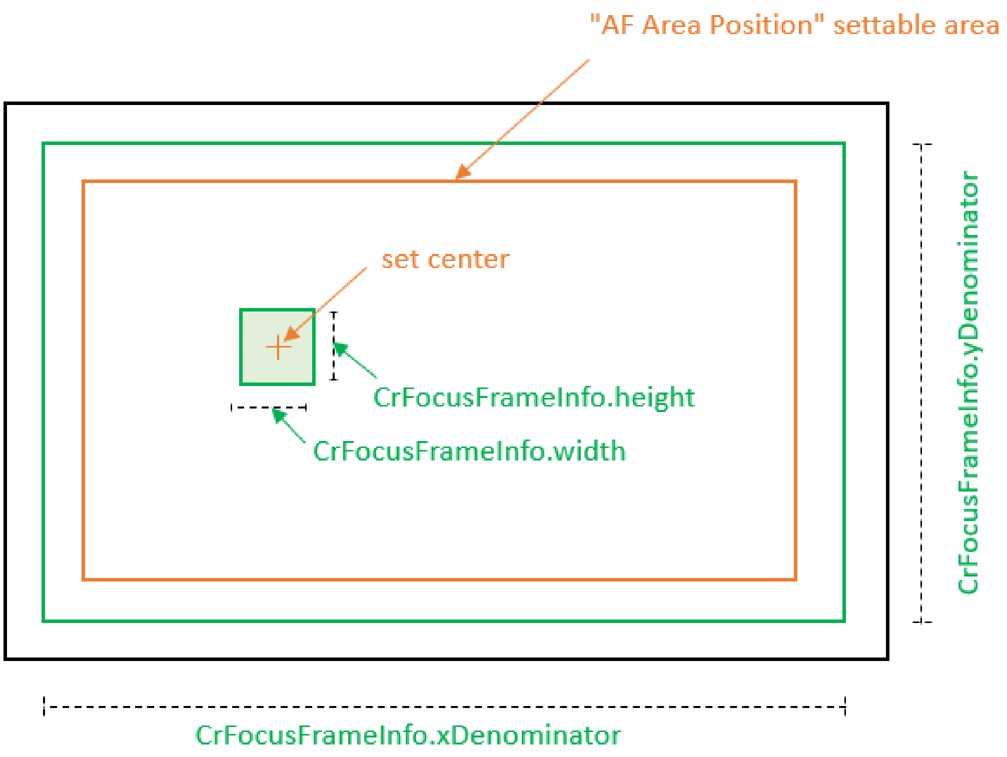
Value	Explanation
-7	min Specify to change the focus to Near. Can be set from -1 to -7 in steps. Larger value makes the movement width larger.
7	max Specify to change the focus to Far. Can be set in steps of 1 to 7. Larger value makes the movement width larger.
1	step

CrDeviceProperty_AF_Area_Position

Set the AF Area Position(x,y)

Value	Explanation
0x00000000～0xFFFFFFFF	Set the center position of the AF frame. The x coordinate is set in the upper two bytes and the y coordinate is set in the lower two bytes The range of X is 0～639 (0x027F), and the range of Y is 0～479 (0x01DF). AF frame size is acquired by CrFocusFrameInfo. CrFocusFrameInfo is in LiveViewProperty. The settable area is more inside by half the frame size than CrFocusFrameInfo.xDenominator/yDenominator. Note: The range in which the coordinates can be specified varies depending on the model, aspect setting, and AF setting.

Fig.Relationship between CrFocusframeInfo and settable area


"AF Area Position" settable area
set center
CrFocusFrameInfo.height
CrFocusFrameInfo.width
CrFocusFrameInfo.yDenominator
CrFocusFrameInfo.xDenominator

CrDeviceProperty_Zoom_Scale

Get/Set the Zoom Scale

Value	Explanation	
Variable	min	min/max/Value should be set in units of “step”. Ex.) min: 1000, max: 8000, step: 200, value: 1200 (min = x1.0, max = x8.0, value = x1.2)
Variable	max	
Variable	step This value varies depending on the camera's configurable conditions. (in units of 0.001)	

CrDeviceProperty_Zoom_Setting

Get/Set the Zoom Setting

Parameter Code	Explanation
CrZoomSetting_OpticalZoomOnly	Optical zoom only
CrZoomSetting_SmartZoomOnly	Smart zoom only
CrZoomSetting_On_ClearImageZoom	Clear image zoom on
CrZoomSetting_On_DigitalZoom	Digital zoom (and Clear image zoom) on

CrDeviceProperty_Zoom_Operation

Execute the Zoom Operation

Parameter Code	Explanation
CrZoomOperation_Wide	Zoom out (-) When you specify zoom out, the zoom out continues until it stops or until the lens or setting limit is reached.
CrZoomOperation_Stop	Specifies when to stop zooming in / out.
CrZoomOperation_Tele	Zoom in (+) When you specify zoom in, the zoom in continues until it stops or until the lens or setting limit is reached.

CrDeviceProperty_Movie_File_Format

Get/Set the File Format(Movie)

Parameter Code	Explanation
CrFileFormatMovie_AVCHD	AVCHD
CrFileFormatMovie_MP4	MP4
CrFileFormatMovie_XAVC_S_4K	XAVC S 4K
CrFileFormatMovie_XAVC_S_HD	XAVC S HD
CrFileFormatMovie_XAVC_HS_4K	XAVC HS 4K
CrFileFormatMovie_XAVC_S_L_4K	XAVC S-L 4K
CrFileFormatMovie_XAVC_S_L_HD	XAVC S-L HD
CrFileFormatMovie_XAVC_S_I_4K	XAVC S-I 4K
CrFileFormatMovie_XAVC_S_I_HD	XAVC S-I HD

Note: In some models, "XAVC S-L xx" is displayed as "XAVC S xx" in their menu.

CrDeviceProperty_Movie_Recording_Setting

Get/Set the Recording Setting(Movie)

Parameter Code	Explanation
CrRecordingSettingMovie_60p_50M	60p 50M / XAVC S
CrRecordingSettingMovie_30p_50M	30p 50M / XAVC S
CrRecordingSettingMovie_24p_50M	24p 50M / XAVC S
CrRecordingSettingMovie_50p_50M	50p 50M / XAVC S
CrRecordingSettingMovie_25p_50M	25p 50M / XAVC S
CrRecordingSettingMovie_60i_24M	60i 24M(FX) / AVCHD
CrRecordingSettingMovie_50i_24M_FX	50i 24M(FX) / AVCHD
CrRecordingSettingMovie_60i_17M_FH	60i 17M(FH) / AVCHD
CrRecordingSettingMovie_50i_17M_FH	50i 17M(FH) / AVCHD
CrRecordingSettingMovie_60p_28M_PS	60p 28M(PS) / AVCHD
CrRecordingSettingMovie_50p_28M_PS	50p 28M(PS) / AVCHD
CrRecordingSettingMovie_24p_24M_FX	24p 24M(FX) / AVCHD
CrRecordingSettingMovie_25p_24M_FX	25p 24M(FX) / AVCHD
CrRecordingSettingMovie_24p_17M_FH	24p 17M(FH) / AVCHD
CrRecordingSettingMovie_25p_17M_FH	25p 17M(FH) / AVCHD
CrRecordingSettingMovie_120p_50M_1280x720	120p 50M (1280x720) / XAVC S
CrRecordingSettingMovie_100p_50M_1280x720	100p 50M (1280x720) / XAVC S
CrRecordingSettingMovie_1920x1080_30p_16M	1920x1080 30p 16M / MP4
CrRecordingSettingMovie_1920x1080_25p_16M	1920x1080 25p 16M / MP4
CrRecordingSettingMovie_1280x720_30p_6M	1280x720 30p 6M / MP4

CrRecordingSettingMovie_1280x720_25p_6M	1280x720 25p 6M / MP4
CrRecordingSettingMovie_1920x1080_60p_28M	1920x1080 60p 28M / MP4
CrRecordingSettingMovie_1920x1080_50p_28M	1920x1080 50p 28M / MP4
CrRecordingSettingMovie_60p_25M_XAVC_S_HD	60p 25M / XAVC S HD
CrRecordingSettingMovie_50p_25M_XAVC_S_HD	50p 25M / XAVC S HD
CrRecordingSettingMovie_30p_16M_XAVC_S_HD	30p 16M / XAVC S HD
CrRecordingSettingMovie_25p_16M_XAVC_S_HD	25p 16M / XAVC S HD
CrRecordingSettingMovie_120p_100M_1920x1080_XAVC_S_HD	120p 100M (1920x1080) / XAVC S HD
CrRecordingSettingMovie_100p_100M_1920x1080_XAVC_S_HD	100p 100M (1920x1080) / XAVC S HD
CrRecordingSettingMovie_120p_60M_1920x1080_XAVC_S_HD	120p 60M (1920x1080) / XAVC S HD
CrRecordingSettingMovie_100p_60M_1920x1080_XAVC_S_HD	100p 60M (1920x1080) / XAVC S HD
CrRecordingSettingMovie_30p_100M_XAVC_S_4K	30p 100M / XAVC S 4K
CrRecordingSettingMovie_25p_100M_XAVC_S_4K	25p 100M / XAVC S 4K
CrRecordingSettingMovie_24p_100M_XAVC_S_4K	24p 100M / XAVC S 4K
CrRecordingSettingMovie_30p_60M_XAVC_S_4K	30p 60M / XAVC S 4K
CrRecordingSettingMovie_25p_60M_XAVC_S_4K	25p 60M / XAVC S 4K
CrRecordingSettingMovie_24p_60M_XAVC_S_4K	24p 60M / XAVC S 4K
CrRecordingSettingMovie_600M_422_10bit	600M 422 10bit
CrRecordingSettingMovie_500M_422_10bit	500M 422 10bit
CrRecordingSettingMovie_300M_422_10bit	300M 422 10bit
CrRecordingSettingMovie_280M_422_10bit	280M 422 10bit
CrRecordingSettingMovie_250M_422_10bit	250M 422 10bit
CrRecordingSettingMovie_240M_422_10bit	240M 422 10bit
CrRecordingSettingMovie_222M_422_10bit	222M 422 10bit
CrRecordingSettingMovie_200M_422_10bit	200M 422 10bit
CrRecordingSettingMovie_200M_420_10bit	200M 420 10bit
CrRecordingSettingMovie_200M_420_8bit	200M 420 8bit
CrRecordingSettingMovie_185M_422_10bit	185M 422 10bit
CrRecordingSettingMovie_150M_420_10bit	150M 420 10bit
CrRecordingSettingMovie_150M_420_8bit	150M 420 8bit
CrRecordingSettingMovie_140M_422_10bit	140M 422 10bit
CrRecordingSettingMovie_111M_422_10bit	111M 422 10bit
CrRecordingSettingMovie_100M_422_10bit	100M 422 10bit
CrRecordingSettingMovie_100M_420_10bit	100M 420 10bit
CrRecordingSettingMovie_100M_420_8bit	100M 420 8bit

CrRecordingSettingMovie_93M_422_10bit	93M 422 10bit
CrRecordingSettingMovie_89M_422_10bit	89M 422 10bit
CrRecordingSettingMovie_75M_420_10bit	75M 420 10bit
CrRecordingSettingMovie_60M_420_8bit	60M 420 8bit
CrRecordingSettingMovie_50M_422_10bit	50M 422 10bit
CrRecordingSettingMovie_50M_420_10bit	50M 420 10bit
CrRecordingSettingMovie_50M_420_8bit	50M 420 8bit
CrRecordingSettingMovie_45M_420_10bit	45M 420 10bit
CrRecordingSettingMovie_30M_420_10bit	30M 420 10bit
CrRecordingSettingMovie_25M_420_8bit	25M 420 8bit
CrRecordingSettingMovie_16M_420_8bit	16M 420 8bit

CrDeviceProperty_Movie_Recording_FrameRateSetting

Get/Set the Recording Frame Rate Setting(Movie)

Parameter Code	**Explanation**
CrRecordingFrameRateSettingMovie_120p	120p
CrRecordingFrameRateSettingMovie_100p	100p
CrRecordingFrameRateSettingMovie_60p	60p
CrRecordingFrameRateSettingMovie_50p	50p
CrRecordingFrameRateSettingMovie_30p	30p
CrRecordingFrameRateSettingMovie_25p	25p
CrRecordingFrameRateSettingMovie_24p	24p

CrDeviceProperty_Interval_Rec_Mode

Get the Interval REC Mode

Parameter Code	**Explanation**
CrIntervalRecMode_OFF	OFF
CrIntervalRecMode_ON	ON

CrDeviceProperty_Still_Image_Trans_Size

Get/Set the Still Image Trans Size

Parameter Code	**Explanation**
CrPropertyStillImageTransSize_Original	Original

CrPropertyStillImageTransSize_SmallSizeJPEG	Small Size JPEG

CrDeviceProperty_RAW_J_PC_Save_Image

Get/Set the RAW+J PC Save Image

Parameter Code	Explanation
CrPropertyRAWJPCSaveImage_RAWAndJPEG	RAW & JPEG
CrPropertyRAWJPCSaveImage_JPEGOnly	JPEG Only
CrPropertyRAWJPCSaveImage_RAWOnly	RAW Only
CrPropertyRAWJPCSaveImage_RAWAndHEIF	RAW & HEIF
CrPropertyRAWJPCSaveImage_HEIFOnly	HEIF Only

CrDeviceProperty_LiveView_Image_Quality

Get/Set the LiveView Quality

Parameter Code	Explanation
CrPropertyLiveViewImageQuality_Low	Low
CrPropertyLiveViewImageQuality_High	High

CrDeviceProperty_CustomWB_Capture_Standby

Get the Custom WB Capture Standby Operation

Parameter Code	Explanation
CrPropertyCustomWBOperation_Disable	Disable
CrPropertyCustomWBOperation_Enable	Enable

Execute the Custom WB Capture Standby

Parameter Code	Explanation
CrPropertyCustomWBCapture_Up	Up
CrPropertyCustomWBCapture_Down	Down

CrDeviceProperty_CustomWB_Capture_Standby_Cancel

Get the Custom WB Capture Standby Cancel Operation

Parameter Code	Explanation
CrPropertyCustomWBOperation_Disable	Disable
CrPropertyCustomWBOperation_Enable	Enable

Execute the Custom WB Capture Standby Cancel

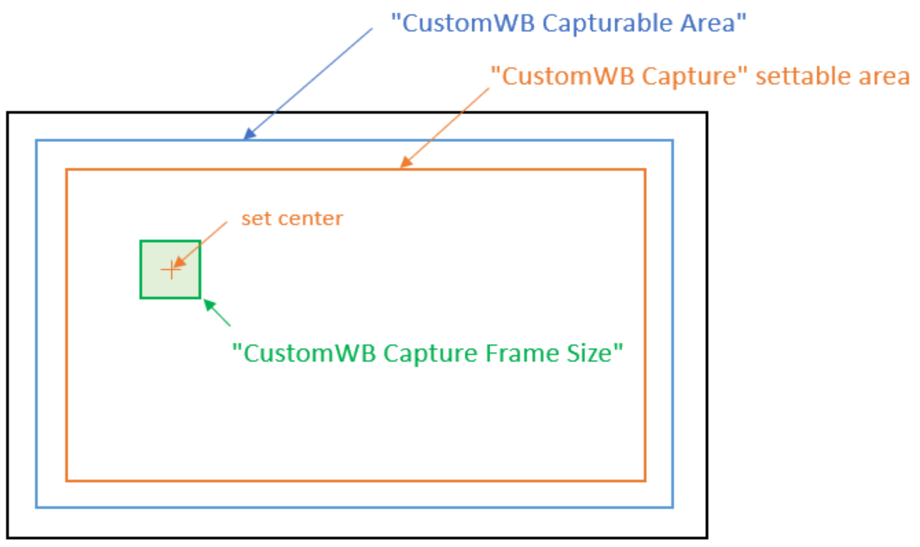
Parameter Code	Explanation
CrPropertyCustomWBCapture_Up	Up
CrPropertyCustomWBCapture_Down	Down

CrDeviceProperty_CustomWB_Capture

Execute the Custom WB Capture

Value	Explanation	
0x00000000	min	The x coordinate is set in the upper two bytes and the y coordinate is set in the lower two bytes The enable range can be obtained from "Custom WB Capturable Area". "Custom WB Capture Frame Size" is currently 64x64 fixed size. The settable area is more inside by half the Frame Size than "Custom WB Capturable Area". Note: The settable range varies depending on the model and aspect setting.
0xFFFFFFFF	max	
1	step	

Fig. Relationship between capture frame size and settable position


"CustomWB Capturable Area"
"CustomWB Capture" settable area
set center
"CustomWB Capture Frame Size"

CrDeviceProperty_SnapshotInfo

Get the Shooting File Info

Value	Explanation	
0x0000	min	0x0000:transferable file doesn't exit 0x0001-0x7FFF:exist file If the value is over than 0x8001(MSB is 0b01), can get the Shot files.
0xFFFF	max	
0x0001	step	

CrDeviceProperty_BatteryRemain

Get the Battery Remaining (%)

Value	Explanation
0xFF(untaken)	min
0x64(100%)	max
0x01	step

CrDeviceProperty_BatteryLevel

Get the Battery Level Indicator

Parameter Code	Explanation
CrBatteryLevel_Fake	Fake Battery
CrBatteryLevel_PreEndBattery	Pre-End Battery
CrBatteryLevel_1_4	Battery Level 1/4
CrBatteryLevel_2_4	Battery Level 2/4
CrBatteryLevel_3_4	Battery Level 3/4
CrBatteryLevel_4_4	Battery Level 4/4
CrBatteryLevel_1_3	Battery Level 1/3
CrBatteryLevel_2_3	Battery Level 2/3
CrBatteryLevel_3_3	Battery Level 3/3
CrBatteryLevel_PreEnd_PowerSupply	Pre-End Battery with USB BusPower Supply
CrBatteryLevel_1_4_PowerSupply	Battery Level 1/4 with USB BusPower Supply
CrBatteryLevel_2_4_PowerSupply	Battery Level 2/4 with USB BusPower Supply
CrBatteryLevel_3_4_PowerSupply	Battery Level 3/4 with USB BusPower Supply
CrBatteryLevel_4_4_PowerSupply	Battery Level 4/4 with USB BusPower Supply
CrBatteryLevel_USBPowerSupply	USB BusPower Supply

CrDeviceProperty_RecordingState

Get the Movie Recording State

Parameter Code	Explanation
CrMovie_Recording_State_Not_Recording	Not Recording
CrMovie_Recording_State_Recording	Recording
CrMovie_Recording_State_Recording_Failed	Recording Failed

CrDeviceProperty_LiveViewStatus

LiveView Status

Parameter Code	Explanation
CrLiveView_Disable	LiveView Support but Disable just now :If this value is set, the host should not get the LiveView Image.
CrLiveView_Enable	LiveView Support and Enable :The host can get the LiveView Image and activate LiveView button if have.
CrLiveView_NotSupport	LiveView Not Support :Just definition, If the camera doesn't support Liveview, the host can't get this property by any operation.

CrDeviceProperty_FocusIndication

Get the Focus Indication

Parameter Code	Explanation
CrFocusIndicator_Unlocked	Unlock
CrFocusIndicator_Focused_AF_S	[AF-S]Focussed, and AF Locked State
CrFocusIndicator_NotFocused_AF_S	[AF-S]Not focussed, and Low Contrast State
CrFocusIndicator_TrackingSubject_AF_C	[AF-C]Tracking Subject motion
CrFocusIndicator_Focused_AF_C	[AF-C]Focussed State
CrFocusIndicator_NotFocused_AF_C	[AF-C]Not focussed, and Low Contrast State

CrDeviceProperty_MediaSLOT1_Status

Get the Media (SLOT1) Status

Parameter Code	Explanation
CrSlotStatus_OK	OK
CrSlotStatus_NoCard	No card
CrSlotStatus_CardError	Card error
CrSlotStatus_RecognizingOrLockedError	Card recognizing/Card locked and DB error

CrDeviceProperty_MediaSLOT1_RemainingNumber

Get the Remaining number shots of Media (SLOT1)

Value	Explanation	
0x00000000	min	Unit is the remaining number of shots.
0xFFFFFFFF	max	
0x00000001	step	

CrDeviceProperty_MediaSLOT1_RemainingTime

Get the Remaining shooting time of Media (SLOT1)

Value	Explanation	
0x00000000	min	Unit is second, the remaining time of movie recording.
0xFFFFFFFF	max	
0x00000001	step	

CrDeviceProperty_MediaSLOT1_FormatEnableStatus

Get the Media Format Enable Status(SLOT1)

Parameter Code	Explanation
CrMediaFormat_Disable	Disable
CrMediaFormat_Enable	Enable

CrDeviceProperty_MediaSLOT2_Status

Get the Media (SLOT2) Status

Parameter Code	Explanation
CrSlotStatus_OK	OK
CrSlotStatus_NoCard	No card
CrSlotStatus_CardError	Card error
CrSlotStatus_RecognizingOrLockedError	Card recognizing/Card locked and DB error

CrDeviceProperty_MediaSLOT2_RemainingNumber

Get the Remaining number shots of Media (SLOT2)

Value	Explanation	
0x00000000	min	Unit is the remaining number of shots.
0xFFFFFFFF	max	
0x00000001	step	

CrDeviceProperty_MediaSLOT2_RemainingTime

Get the Remaining shooting time of Media (SLOT2)

Value	Explanation	
0x00000000	min	Unit is second, the remaining time of movie recording.
0xFFFFFFFF	max	
0x00000001	step	

CrDeviceProperty_MediaSLOT2_FormatEnableStatus

Get the Media Format Enable Status(SLOT2)

Parameter Code	Explanation
CrMediaFormat_Disable	Disable
CrMediaFormat_Enable	Enable

CrDeviceProperty_Media_FormatProgressRate

Get the Media Format Progress Rate

Value	Explanation
0x00000000	Invalid
Other than above values	Progress rate Lower 16bit is denominator, Higher 16bit is molecules. Calculate the progress rate each time. e.g.) 0x003600C8 means 27%. (by the following calculations. (0x36/0xC8) * 100)

CrDeviceProperty_Interval_Rec_Status

Get the Interval REC Status

Parameter Code	Explanation
CrIntervalRecStatus_WaitingStart	Waiting Start
CrIntervalRecStatus_IntervalShooting	Interval Shooting

CrDeviceProperty_CustomWB_Execution_State

Get the Custom WB Execution State

Parameter Code	Explanation
CrPropertyCustomWBExecutionState_Invalid	Invalid
CrPropertyCustomWBExecutionState_Standby	Standby
CrPropertyCustomWBExecutionState_Capturing	Capturing
CrPropertyCustomWBExecutionState_OperatingCamera	OperatingCamera

CrDeviceProperty_CustomWB_Capturable_Area

Get the Custom WB Capturable Area(x,y)

Value	Explanation	
0x00000000	min	The device can get the capturable area of Custom WB Capturing with this property. The x coordinate is set in the upper two bytes and the y coordinate is set in the lower two bytes This value varies depends on the model and aspect setting. e.g.) min 0x00200020 means TopLeft=32,32.
0xFFFFFFFF	max	
0x00000001	step	

CrDeviceProperty_CustomWB_Capture_Frame_Size

Get the Custom WB Capture Frame Size(x,y)

Value	Explanation	
0x00000000	min	The frame width is set in the upper two bytes and the frame height is set in the lower two bytes This value is currently 0x00400040 (64x64) fixed.
0xFFFFFFFF	max	
0x00000001	step	

CrDeviceProperty_CustomWB_Capture_Operation

Get the Custom WB Capture Operation Enable Status

Parameter Code	Explanation
CrPropertyCustomWBOperation_Disable	Disable
CrPropertyCustomWBOperation_Enable	Enable

CrDeviceProperty_Zoom_Operation_Status

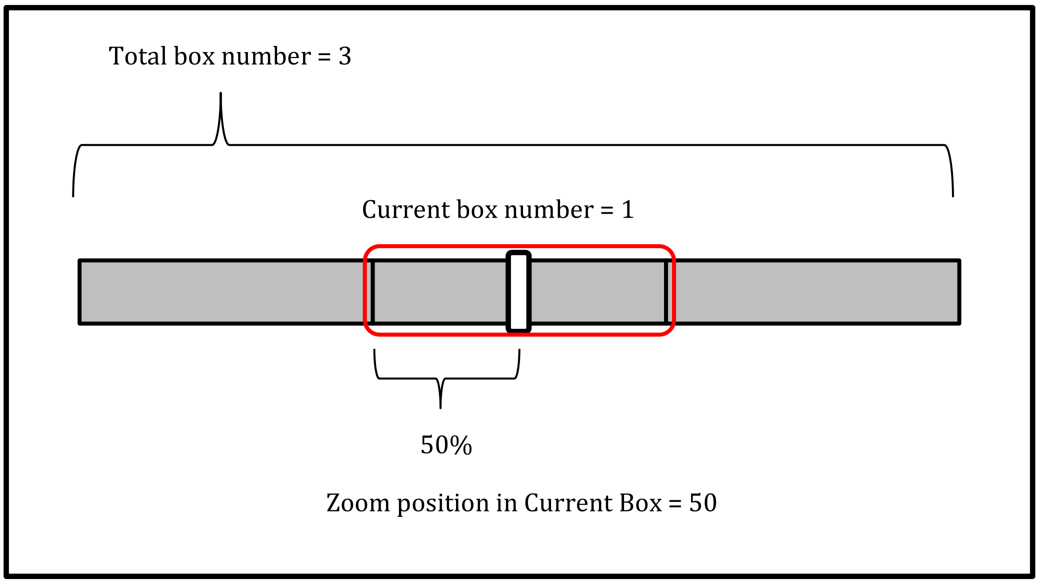
Get the Zoom Operation Enable Status

Parameter Code	Explanation
CrZoomOperationEnableStatus_Disable	Disable
CrZoomOperationEnableStatus_Enable	Enable

CrDeviceProperty_Zoom_Bar_Information

Get the Zoom Bar Information

Value		Explanation
31-24bit		Total box number
	0	min
	0xFF	max
	1	step
23-16bit		Current box number
	0	min
	0xFF	max
	1	step
15- 0bit		Zoom position in Current Box
	0x00	min
	0x64	max
	0x01	step



CrDeviceProperty_Zoom_Type_Status

Get the Zoom Type Status

Parameter Code	Explanation
CrZoomTypeStatus_OpticalZoom	Optical zoom only
CrZoomTypeStatus_SmartZoom	Smart zoom only
CrZoomTypeStatus_ClearImageZoom	Clear image zoom
CrZoomTypeStatus_DigitalZoom	Digital zoom

Tips / Trouble Shooting

Shutter Release

If you struggle to make "Shutter Release" success in a remote control, please try to set camera settings "Exposure Program Mode" with "M(Manual)" and "FocusMode" with "MF(Manual Focus)".
∵As camera accepts "Shutter release control" after coming into focus in several Auto Focus modes, sometimes focus mode setting, focus area setting, and shooting environmental conditions prevent camera to accept "Shutter Release".

Remote Control Settings Example

1. "CrDeviceProperty_PriorityKeySettings" with "CrPriorityKey_PCRemote"
2. "CrDeviceProperty_ExposureProgramMode" with "CrExposure_M_Manual"
3. "CrDeviceProperty_FocusMode" with "CrFocus_MF"
4. "CrCommandId_Release" with "CrCommandParam_Down"
5. "CrCommandId_Release" with "CrCommandParam_Up"

Also, memory card full situation prevents shutter release from execution, so it is recommended to prepare enough space in the memory card and / or prepare dual memory cards before remote control.

Shutter Half Release / Auto Focus

If you struggle to make "Shutter Half Release" success and come into focus successfully in remote controls, please try to set camera settings "FocusMode" with "AF-S", and "FocusArea" with "Wide".
∵As camera occasionally takes time relatively to come into focus depends on settings and shooting environmental conditions in several auto focus modes, above settings have relatively wide acceptance to come into focus.

Remote Control Settings Example

1. "CrDeviceProperty_PriorityKeySettings" with "CrPriorityKey_PCRemote"
2. "CrDeviceProperty_FocusMode" with "CrFocus_AF_S"
3. "CrDeviceProperty_FocusArea" with "CrFocusArea_Wide"
4. "CrDeviceProperty_S1" with "CrLockIndicator_Locked"
5. "CrDeviceProperty_S1" with "CrLockIndicator_Unlocked"

Manual Focus

If you struggle to control focus manually in remote controls, please try to set camera settings "FocusMode" with "MF(Manual Focus)".

Remote Control Settings Example

1. "CrDeviceProperty_PriorityKeySettings" with "CrPriorityKey_PCRemote"
2. "CrDeviceProperty_FocusMode" with "CrFocus_MF"

Device Property

If you struggle to change camera settings, it is recommended to check enable flag in each DeviceProperty by sending GetDeviceProerty and receiving the latest information before sending SetDeviceProperty.
∵As the specification of camera products, camera settings have exclusive conditions. For example, focus control Near/Far is not acceptable in Focus Mode "AF-S". In order to identify whether an issue is coming from remote control related or camera settings acceptable/unacceptable conditions, you better try what you want to do first w/o remote control but w/ direct camera operation by camera buttons / menu settings. Then copy operations with remote control. "Help Guide" for each product may help you to understand the specification of camera products including acceptable/unacceptable conditions of settings.

Remote Control Settings Example

1. "GetDeviceProperties" with "CrDevicePropertyCode"
2. Check "CrPropertyEnableFlag" of "CrDeviceProperty"
3. "SetDeviceProperties" with "CrDevicePropertyCode"

Also, it is recommended to set a value from candidate values list in each DeviceProperty after sending GetDeviceProerty and receiving the latest information before sending SetDeviceProperty.
∵As the specification of camera products, camera settings have variable acceptance for value depends on settings and shooting environmental conditions. For example, acceptable F number value varies depends on the lens attached to the camera, other settings, and the shooting environmental conditions.

Remote Control Settings Example

1. "GetDeviceProperties" with "CrDevicePropertyCode"
2. Check "valuesSize" and "values" of "CrDeviceProperty"
3. "SetDeviceProperties" with "CrDevicePropertyCode"

Some of DeviceProperties are originally assigned on HardKeys of the camera product, and in these cases,need to change KeyPriority from "CameraPosition" to "PCRemote" before sending SetDeviceProperty. This applies to "ExposureProgramMode", "FocusMode" and "Still Capture Mode(Drive Mode)".

Remote Control Settings Example

1. "CrDeviceProperty_PriorityKeySettings" with "CrPriorityKey_PCRemote"
2. "SetDeviceProperties" with "CrDevicePropertyCode"

Transfer of shot images preparation

If you struggle to transfer shot images to PC, please check if you changed "StillImageStoreDestination" before shutter button release. You can select from HostPC/MemoryCard/HostPCAndMemoryCard. When you transfer shot images to PC, need to change it to HostPC/HostPCAndMemoryCard beforehand.

Remote Control Settings Example

1. "CrDeviceProperty_StillImageStoreDestination" with "CrStillImageStoreDestination_HostPCAndMemoryCard(or _HostPC)"
2. "CrCommandId_Release" with "CrCommandParam_Down"
3. "CrCommandId_Release" with "CrCommandParam_Up"
4. Check the folder set by SetSaveInfo() and open image files transferred to PC.

Please note that if once Host PC transfer is set like above, camera side also starts preparing and sending out image files, it is recommended to disconnect after finishing transfer of all images shot on the camera. If disconnected before transfer finishes, camera and PC restart to transfer after reconnection, except for camera power off or physical disconnection case.

Selected Media Format

If Still Image Save Destination is Host Device, recording media cannot be initialized.

If you want to initialize it, change Still Image Save Destination to Camera or Host Device and Camera.

Remote Control Settings Example

1. "CrDeviceProperty_StillImageStoreDestination" with "CrStillImageStoreDestination_HostPCAndMemoryCard(or _MemoryCard)"

Zoom Operation

Zoom Operation is available only with Power Zoom Lenses such as SELP1650, SELP18105G, SELP18110G, SELP18200 and SELP28135G. You can also control Clear Image Zoom/Digital Zoom only when you attached the Power Zoom Lens to your camera.(Please note that you need to change zoom setting of camera body menu).

Live View

If you struggle to have stable live view images, please check following factors affect to transmission of LiveView images.

-Traffic on the physical connection between PC and camera, such as HUB connection, not related devices connection, and so on.

-Traffic on the communicational connection between PC and camera, such as frequent shutter releases and transfers, frequent Get/Set device properties, and so on.

-Performance of PC (CPU power, memory resource, device specification, etc).

-Some functions to be disabled they can be processing loads to CPU on the Single Board Computer, such as WiFi function.

If you prefer stable frame rate of live view images, minimizing image size of Live View images (and/or capturing images), reducing frequency of shutter release, stopping capturing images, and stopping transferring images to PC contributes to it.

Camera Settings Saving

After changing camera settings, if you detach a battery from a camera (or stop power supply through power supply cable) without completing power off sequence with camera power button control, there is no guarantee that camera setting changes are saved. It is recommended to complete power off sequence with camera power button control at least once after you change camera settings, if you prefer to resume camera settings as you changed for next use.

Focus Magnifier Setting

If you want to update "Focus Magnifier Setting", implement the following steps.
refs. Device Properties and Live View Properties

1. Get a list of properties using the GetDeviceProperties
2. Look for "Focus Magnifier Setting" from the list of properties to find out the list of selectable focus magnification

Example:

```
             switch (property->GetCode()) {

             case CrDevicePropty_Focus_Magnifier_Setting:

                          CrInt64u currentvalue = static_cast<CrInt64u>(property-
>GetCurrentValue());

                          CrInt32u ratioNow = (currentvalue >> 32);

                          CrInt16u xNow = ((currentvalue >> 16) & 0xFFFF);

                          CrInt16u yNow = (currentvalue & 0xFFFF);

                          CrInt32u valCount = property->GetValueSize() /
sizeof(CrInt64u);

                          CrInt64u* ratioSetList = new CrInt64u[valCount];

                          memcpy(ratioSetList, property->GetValues(),(size_t)property-
>GetValueSize());
```

3. Use the GetLiveViewProperties to get a list of Live View properties
4. Look for "CrMagPosInfo" in the retrieved list of Live View properties to find out the range of configurable positions

Example:

```
             switch (lvproperty->GetCode()) {

             case CrDeviceLiveViewProperty_Focus_Magnifier_Position:

                          if (CrFrameInfoType::CrFrameInfoType_Magnifier_Position ==
lvproperty->GetFrameInfoType()) {

                                       CrMagPosInfo *pPosInfo

= (CrMagPosInfo*)(lvproperty->GetValue());

                                       posXmax = pPosInfo->xDenominator;

                                       posYmax = pPosInfo->yDenominator;
```

5. Create a 64 bit value by combining the magnification rate obtained in step 2 and the coordinates that do not exceed the range obtained in step 4
6. Call SetDeviceProperty with the value you created in step 5

```
Example:

            CrInt32u setX = 200; // Between 0 and (posXmax-1)

            CrInt32u setY = 150; // Between 0 ant (posYmax-1)

            CrInt64u setvalue = (ratioSetList[2] & 0xFFFFFFFF00000000) | (setX << 16)
| setY;

            CrDeviceProperty prop;

            prop.SetCode(CrDeviceProperty_Focus_Magnifier_Setting);

            prop.SetCurrentValue(setvalue);

            prop.SetValueType(CrDataType_UInt64);

            SetDeviceProperty(deviceHandle, &prop);
```

More information

Trademarks and acknowledgements