PowerShot

RemoteCapture

Software Development Kit

Software Developer's Guide

Ver 1.1.0c

Revision History

Document Version	Issued	Updated Content
1.1	2005/12/09	First edition.
1.1.0a	2006/06/02	Addition of supported camera models.
1.1.0b	2006/11/08	Addition of supported camera models.
1.1.0c	2007/07/30	Addition of supported camera models. The specification of the device property "prPTP_DEV_PROP_CAPTURE_TRANSFER_MODE" is changed.

Table of Contents

1 Getting Started

The PS-ReC SDK (Power Shot Remote Capture SDK) is a software development kit. The PS-ReC SDK provides an interface for controlling Canon digital cameras from a computer to capture images.

The PS-ReC SDK is provided as a library that can be linked to application software and programs.

This chapter contains the following sections:

- Introduction
- System Requirements
- Configuration of the PS-ReC SDK
- Setting Up a Project
- How to Use PS-ReC SDK Modules

Introduction

The PS-ReC SDK is a library that provides an interface for controlling Canon digital cameras from a computer to capture images remotely.

Applications developed using the PS-ReC SDK allow for implementation of various functions, such letting the user operate the shutter of a camera connected to a computer to capture still images.

Note:	The PS-ReC SDK does not provide an interface to access image data captured by Canon digital cameras or competing digital cameras.

Contents of the PS-ReC SDK Package

The PS-ReC SDK package contains the items shown below.

- Media for installing the PS-ReC SDK
- Software Developer's Guide (this manual)

System Requirements

This section describes the system requirements for using the PS-ReC SDK.

Supported Camera Models

This version of the PS-ReC SDK supports the camera models shown below.

PowerShot A620,
PowerShot S80,
PowerShot S3 IS
PowerShot G7
PowerShot A640
PowerShot S5 IS

Note 1: Some camera models do not support a remote capture function. Remote capture is possible from an application using the PS-ReC SDK only when the connected camera supports a remote capture function.

Note 2: The PS-ReC SDK does not support wireless connection.

For older camera models not listed above, use the CD-SDK (Canon Digital Camera Software Development Kit) Ver. 7.3.

Development System Environment

Host computer

Minimum configuration:

- Pentium or higher processor
- At least 64 MB of RAM (except Windows 2000 SP4/XP)
  At least 128 MB of RAM (Windows 2000 SP4/XP)
- 800 x 600 pixel, 256 color (8 bit) or higher video adapter and monitor

Recommended configuration:

- 500 MHz or higher Pentium processor
- At least 128 MB of RAM (except Windows 2000 SP4/XP)
  At least 256 MB of RAM (Windows 2000 SP4/XP)
- 1024 x 768 pixel, High Color (16 bit) or higher video adapter and monitor

Operating system (OS)

- Windows 98SE, Windows Me, Windows 2000 SP4, Windows XP

Development environment

- Microsoft Visual C++ 6.0

Note 1: Not supported on the Macintosh.

Target System Environment

Not only must the camera be supported by this version of the PS-ReC SDK, but a target system that meets the following requirements is also necessary to run the client application created by the PS-ReC SDK.

Host computer

Minimum configuration:

- Pentium or higher processor
- At least 64 MB of RAM (except Windows 2000 SP4/XP)
  At least 128 MB of RAM (Windows 2000 SP4/XP)
- 800 x 600 pixel, 256 color (8 bit) or higher video adapter and monitor

Recommended configuration:

- 500 MHz or higher Pentium processor
- At least 128 MB of RAM (except Windows 2000 SP4/XP)
  At least 256 MB of RAM (Windows 2000 SP4/XP)
- 1024 x 768 pixel, True Color (24 bit) or higher video adapter and monitor

Operating system (OS)

- Windows 98SE, Windows Me, Windows 2000 SP4, Windows XP

Note: Not supported on the Macintosh.

Configuration of the PS-ReC SDK

To use the PS-ReC SDK, copy the entire PSReCSDK folder to a desired folder on your computer.

For example, copying the PSReCSDK folder to C:\ will create a development environment for using the SDK, as follows.

```
C:\
|
+ - PSReCSDK
      |
      +-- INC : Header file recording folder
      |         PRAPI.h
      |         PRFuncType.h
      |         PRType.h
      |         PRError.h
      |
      +-- LIB : Static library file recording folder
      |         PRSDK.LIB
      |
      +-- REDIST : Dynamic link library (DLL) recording folder
                PRSDK.DLL
                PRLIB.DLL
```

Setting Up a Project

After you have configured the PS-ReC SDK development environment, the next step is to include several header files required by the PS-ReC SDK in the source code of the software you are going to create and then link the static libraries required by the PS-ReC SDK to your project.

PS-ReC SDK header files

To use the PS-ReC SDK, the following four header files found in the \INC folder are necessary.

- PRAPI.h
- PRType.h
- PRError.h
- PRFuncType.h

You must add the path to the folder where the PS-ReC SDK header files are saved (for example "C:\PSReCSDK\INC") in the Include path for your application project work space. Of the four files above, if you include only the PRAPI.h file in your source code, all of the other header files required for the PS-ReC SDK will be included automatically.

Linking to Libraries

In addition to including the header files, you must also link the PS-ReC SDK library to the project.

Link the PRSDK.lib file in the \LIB folder (for example, "C\PSReCSDK\LIB") to the project.

Reference:

You can also use the following methods to link the library provided by the PS-ReC SDK.

1. Load PRSDK.dll using the LoadLibrary function.
2. Retrieve the PR_GetFunctions function addresses using GetProcAddress.
3. Execute the PR_GetFunctions function to retrieve the addresses for the functions provided by the PS-ReC SDK.

How to Use PS-ReC SDK Modules

You must install the modules required to execute the PS-ReC SDK on the same computer where the application that uses the PS-ReC SDK is to be executed. The following explains how to set up an environment where the application can use the PS-ReC SDK, by using the installer for the application.

Copy the PS-ReC SDK modules

Copy to the execution folder of the application that uses the PS-ReC SDK all modules (DLLs) in the \Redist folder in the PS-ReC SDK installation folder (for example, "C\PSReCSDK\Redist").

Note 1: If more recent versions of the modules already exist in the folder you are copying to, be sure not to overwrite them with the older modules. This means that you must compare file versions as you copy.

Note 2: You must never copy the PS-ReC SDK modules into the Windows system folder or the Windows folder. Be sure to copy them into the application folder.

To connect to a camera

To connect the application to a camera, you need driver software (such as a TWAIN driver or WIA driver) that connects the camera to a computer (assuming that it is running Windows 98 SE/Me/ 2000 SP4). Driver software is either included with the digital camera package or can be obtained as a camera accessory. You can also download driver software from the Canon web site (with the exception of certain regions).

Driver software is not required on computers running Windows XP. On these computers, use the standard driver provided by Windows.

The PS-ReC SDK connects a camera and communicates with it via driver software that supports Canon digital cameras. Therefore, the computer and camera must be connected properly by driver software.

2 Overview of the Canon Digital Camera PS-ReC SDK

The Canon Digital Camera PS-ReC SDK consists of a group of C interface functions. When using a feature provided by the PS-ReC SDK, a PS-ReC SDK client (such as an application or DLL created using the PS-ReC SDK) calls the corresponding PS-ReC SDK function.

This chapter provides a basic overview of the PS-ReC SDK functions. It is divided into the following sections:

- Remote Capture
- Camera Devices
- PS-ReC SDK Clients
- Non Support of Enumeration of and Access to Captured Data

Remote Capture

Remote Release Control

The main purpose of the PS-ReC SDK is to provide a function to capture still images by operating, from the client you are creating, a camera connected to the computer. This function is called “remote release control" in the PS-ReC SDK.

The captured image data can be transferred to the client application on the computer or saved to a memory card inside the camera. Using PS-ReC SDK, the same processing can take place when the camera's shutter button is pressed..

Viewfinder

You can continuously retrieve live image data from the camera. By displaying the captured images continuously, you can effectively achieve the same view function provided by the viewfinder in the camera.

Camera Devices

Enumeration and Specification of Camera Devices

If the computer is properly connected to Canon digital cameras, the PS-ReC SDK provides the client with a means for enumerating the connected cameras and communicating with a selected camera.

In other words, the client first enumerates all camera devices currently connected to the computer using the functions provided by the PS-ReC SDK.

From these camera devices, the client selects the camera to be accessed based on the device information retrieved during the enumeration process, and then creates a handle (camera handle) for the camera device using the functions provided by the PS-ReC SDK.

Once a camera handle is created, the client can connect to the camera or perform remote release control on the camera by simply passing the camera handle to the functions provided by the PS-ReC SDK, without having to perform detailed settings regarding the camera model or port to which the camera is connected.

In this case, depending on how the camera and computer are connected, the camera may be handled as an exclusive system resource. This means that a single process can simultaneously communicate with multiple cameras connected to different ports, but only one process can access only one camera at a time. If two or more processes access a single camera, operation of the PS-ReC SDK cannot be guaranteed.

Note : The camera device enumeration function of the PS-ReC SDK only enumerates the camera models supported by the PS-ReC SDK as explained in Chapter 1. To enumerate Canon digital cameras not supported by the PS-ReC SDK, use the CD-SDK.

Device Properties

The internal memory of each Canon digital camera stores data indicating the condition, capability, and settings of the camera. These data are called "device properties."

Device properties are used for referencing the various functions of a camera, or for referencing or changing the camera settings. For example, device properties are used when the client specifies the shutter speed, aperture and other settings required for capturing images.

The client can read device properties individually or change certain properties by using the PS-ReC SDK functions that access device property data. To access camera properties, the client and camera must be connected by the PS-ReC SDK.

With the PS-ReC SDK, camera properties can be read and written the same way for all camera models. However, the types of camera properties that are supported may differ depending upon the camera model.

Camera Events

The PS-ReC SDK provides a means for the client to get event notifications concerning cameras and camera-related events.

The client registers in the PS-ReC SDK the callback functions provided in the client. When an event occurs on a camera, the PS-ReC SDK receives an event notification from the camera. When the PS-ReC SDK receives an event notification, it executes the applicable callback function registered by the client. The event callback function is executed with parameters that contain information relating to the event type, and, if available, data specific to the event.

PS-ReC SDK Clients

PS-ReC SDK clients are assumed to be applications and static/dynamic link libraries. A client cannot be a hardware partition handler or any type of low-level system software that interferes with camera driver functions.

A client can execute its own processes, its own threads, or other higher-level client processes and threads. A client can also be a multi-threading application.

Non Support of Enumeration of and Access to Captured Data

The CD-SDK provided a means for enumerating (collecting) data captured by Canon digital cameras, such as still images, movies and sounds, and accessing these data. However, the PS-ReC SDK does not provide a means for enumerating or accessing captured data. To access the storage memory in a camera, use the following interfaces provided by Windows.

- Windows 98SE/Me/2000 SP4: Still Image (STI) interface
- Windows XP. Windows Image Acquisition (WIA) interface

For details on the STI and WIA interfaces, refer to the Help for "Microsoft Platform SDK" or "Windows Driver Development Kit" provided by Microsoft. The relationship of the PS-ReC SDK and STI/WIA interfaces is also explained using a sample in Chapter 7, "Sample."

Similarly, the PS-ReC SDK does not provide a means for enumerating or accessing data files in a memory card that has been mounted as a drive to the host computer's file system using an appropriate memory card adapter. To access a storage device mounted to the computer, use the interface provided by the CD-SDK. Also use the CD-SDK to enumerate or access data files in the memory card of any Canon digital camera not supported by the PS-ReC SDK.

For the CD-SDK, refer to "Canon Digital Camera Software Development Kit Software Developer's Guide." The sample programs that come with the PS-ReC SDK medium explain the situations where you should use the PS-ReC SDK or CD-SDK.

3 Function References

This chapter describes technical references on the interface functions provided by the PS-ReC SDK.

It is divided into the following sections:

- Basic Data Types
- Errors
- Basic Functions
- Basic Camera Device Functions
- Remote Release Control Functions
- Device Property Functions

Note: Some areas of the function specification of the PS-ReC SDK relate to the STI/WIA interfaces of Windows. For these interfaces, you can refer to the following documents:

. " Microsoft Platform SDK " (Microsoft)

. " Windows Driver Development Kit " (Microsoft)

Basic Data Types

This section defines the basic data types used in the PS-ReC SDK. These data types may change in the future due to changes in operating systems and development environments.

```
typedef void                 prVoid;
typedef unsigned char        prUInt8;
typedef          char        prInt8;
typedef          char        prChar;
typedef unsigned short       prWChar;
typedef unsigned short       prUInt16;
typedef         short        prInt16;
typedef unsigned long        prUInt32;
typedef         long         prInt32;
typedef         float        prFloat32;
typedef unsigned __int64     prUInt64;
typedef prUInt32             prResponse;
typedef prUInt16             prBoolean;
typedef prUInt32             prEventID;
typedef prUInt32             prContext;
typedef HWND                 prHWND;
typedef prUInt32             prHandle;
typedef prUInt32             prObjectHandle;
```

Errors

The PS-ReC SDK reports all function errors to the client as function return values.

If a function has been executed successfully, the function returns "prOK (0)" as a return value.

If the function has failed, an error code corresponding to the type of error is returned as a return value. Information indicating the module or component where the error occurred and the cause is included in the error code using bit assignment.

With the PS-ReC SDK, you can also retrieve the module information and cause from the error code, by means of masking with the definition values shown below.

```
#define prERROR_COMPONENTID_MASK    0x00F00000L
/*For component ID mask indicating the component in which
the error occurred*/
#define prERROR_ERRORID_MASK        0x0000FFFFL
/*For error ID mask indicating the type of error*/
```

The component IDs are classified into the following types.

Error component IDs	Description
prERROR_PTP_COMPONENTID	PTP operation errors (Errors returned by the camera's firmware)
prERROR_PRSDK_COMPONENTID	Errors returned by the PS-ReC SDK library
prERROR_WIA_STI_COMPONENTID	Errors generated by the Windows WIA/STI
prERROR_WINDOWS_COMPONENTID	Errors generated by the GetLastError() function in WIN32 API
prERROR_COMIF_COMPONENTID	Windows COM interface errors

Refer to "prError.h" for detailed information about error values.

In the function descriptions provided in this chapter, only the error IDs are specified.

Basic Functions

This section describes the basic functions that are available for use in the PS-ReC SDK. Some of these functions must always be executed. Be aware that problems, such as memory leaks, might occur if they are not executed.

➢ PR_StartSDK

Starts the PS-ReC SDK. When starting the PS-ReC SDK, this function must be called.

When this function is called, an initialization process needed for the PS-ReC SDK to operate, which includes allocating the necessary memory, will be performed.

Syntax

```
prCAPI  PR_StartSDK(
     prVoid
);
```

Error ID

Error ID	Description
prNOT_SUPPORTED	This OS is not supported by the PS-ReC SDK.
prMEM_ALLOC_FAILED	Insufficient memory error.
prINTERNAL_ERROR	An internal error.
Other	Error value of GetLastError() in Win32 API, or STI/WIA interface error value.

Reference:

Note 1: You cannot start two or more PS-ReC SDKs with the same process.

➢ PR_FinishSDK

Ends the PS-ReC SDK. When ending the PS-ReC SDK, this function must be called.

When this function is called, a termination process will be performed, which includes releasing the memory allocated after PR_StartSDK was called. If this function is not called, memory leaks or other problems may occur.

Syntax

```
prCAPI PR_FinishSDK(
     prVoid
);
```

Error ID

Error ID	Description
prINVALID_FN_CALL	PR_StartSDK() has not been called.
Other	Error value of GetLastError() in Win32 API.

➢ PR_GetDllsVersion

Retrieves version information of each PS-ReC SDK module (DLL, etc.) currently used. –

Syntax

```
prCAPI PR_GetDllsVersion(
     prUInt32*           pBufferSize,
     prDllsVerInfo*      pDllVersion
);
```

Parameter

Parameter	Description
pBufferSize	[in/out] Specifies the buffer size of the next parameter. If the function has been executed successfully, the data size stored in the buffer is set.
pDllVersion	[out] Specifies the pointer indicating the buffer. The retrieved version information is stored in accordance with the structure type of prDllsVerInfo.

Error ID

Error ID	Description
prINVALID_PARAMETER	The specified parameter is invalid.
prMEM_ALLOC_FAILED	Insufficient memory error.
prINSUFFICIENT_BUFFER	The buffer size specified by the parameter is smaller than the actual data size. Or, the buffer pointer is NULL.
Other	Error value of GetLastError() in Win32 API.

Reference:

If the pDllVersion buffer size specified by the pBufSize parameter is smaller than the actual data size of the version information to be retrieved, the actual data size will be set in pBufSize and prINSUFFICIENT_BUFFER will be returned. The same applies when pDllVersion is a NULL pointer. If pBufSize is greater than the actual data size, all data will be stored in the buffer, and the stored data size will be set in pBufSize.

For information about the prDllsVerInfo structure, refer to Chapter 4, "Structure and Callback Function Definitions" or "prType.h."

➢ PR_GetFunctions

Retrieves all function pointers provided by the PS-ReC SDK other than PR_GetFunctions().

Syntax

```
cdCAPI PR_GetFunctions(
     prFunctions*    pFunctions
);
```

Parameter

Parameter	Description
pFunctions	[out] Specifies the pointer indicating the buffer. The function pointers provided by the PS-ReC SDK are stored in accordance with the structure type of prFunctions. Allocate memory for the number of prFunctions structures, set prCURRENT_FUNCTABLE_VERSION in the Version member variable of the prFunctions structure, and then specify this parameter.

Error ID

Error ID	Description
prINVALID_PARAMETER	The specified parameter is invalid.
prINCOMPATIBLE_VERSION	Invalid prFunctions structure version.

Reference:

For information about the prFunctions structure, refer to Chapter 4, "Structure and Callback Function Definitions" or "prFuncType.h."

Basic Camera Device Functions

This section describes the basic functions relating to camera devices. Specifically, the following items are described:

- Enumeration of Camera Device Models
- Creation/Deletion of Camera Handles
- Connection/Disconnection of Camera Devices
- Retrieval of Camera Events
- Retrieval of Basic Camera Device Information

Enumeration of Camera Devices

Information regarding the camera devices supported by the PS-ReC SDK can be enumerated using the function described below.

➢ PR_GetDeviceList

Enumerates information regarding the camera devices currently connected to the PC.

Syntax

```
prCAPI PR_GetDeviceList(
     prUInt32*      pBufferSize,
     prDeviceList*  pDeviceList
);
```

Parameter

Parameter	Description
pBufferSize	[in/out] Specifies the buffer size of the next parameter. If the function has been executed successfully, the data size stored in the buffer is set.
pDeviceList	[out] Specifies the pointer indicating the buffer. The enumerated camera device information is stored in the structure type of prDeviceList.

Error ID

Error ID	Description
prINVALID_FN_CALL	PR_StartSDK() has not been called.
prINVALID_PARAMETER	The specified parameter is invalid.
prMEM_ALLOC_FAILED	Insufficient memory error.
prINSUFFICIENT_BUFFER	The buffer size specified by the parameter is smaller than the actual data size. Or, the buffer pointer is NULL.
prINTERNAL_ERROR	An internal error.
Other	Error value of GetLastError() in Win32 API, or STI/WIA interface error value.

Reference:

If the pDeviceList buffer size specified by the pBufSize parameter is smaller than the actual data size of the camera device information to be retrieved, the actual data size will be set in pBufSize and prINSUFFICIENT_BUFFER will be returned. The same applies when pDeviceList is a NULL pointer. If pBufSize is greater than the actual data size, all data will be stored in the buffer, and the stored data size will be set in pBufSize.

Information of each camera device is indicated in the structure type of prDeviceInfoTable. The prDeviceList structure conforms to the array type of prDeviceInfoTable structure. For information about the prDeviceList structure and prDeviceInfoTable structure, refer to Chapter 4, "Structure and Callback Function Definitions" or "prType.h."

Creation/Deletion of Camera Handles

The following section describes the functions used to select, as the connection target, one type of camera device information (prDeviceInfoTable structures) enumerated by PR_GetDeviceList(), and create a camera handle from the selected camera device information or delete the created camera handle.

➢ PR_CreateCameraObject

Creates a camera handle from camera device information. After this function, use the camera handle whenever functions relating to the camera device are used, such as when connecting the camera device or performing remote release control on the device.

Syntax

```
prCAPI  PR_CreateCameraObject(
     prDeviceInfoTable*        pDeviceInfo,
     prHandle*                 pCameraHandle
);
```

Parameter

Parameter	Description
pDeviceInfo	[in] Specifies the camera device information to be processed.
pCameraHandle	[out] A camera handle corresponding to the camera device information in argument 1 is set.

Error ID

Error ID	Description
prINVALID_FN_CALL	PR_StartSDK() has not been called.
prINVALID_PARAMETER	The specified parameter is invalid.
prMEM_ALLOC_FAILED	Insufficient memory error.
Other	Error value of GetLastError() in Win32 API.

Reference:

Note: In the pDeviceInfo parameter, specify the camera device information retrieved by PR_GetDeviceList() without changing its contents in PR_CreateCameraObject(). If a camera handle is created after changing the camera device information in the client, any subsequent use of the new camera handle to connect the camera will generate an error.

➢ PR_DestroyCameraObject

Deletes the camera handle created by PR_CreateCameraObject(). If a created camera handle has become no longer necessary, be sure to delete it using this function. If the camera handle is not deleted, memory leaks or other problems may occur.

Syntax

```
prCAPI  PR_DestroyCameraObject(
     prHandle   CameraHandle
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle to be deleted.

Error ID

Error ID	Description
prINVALID_FN_CALL	PR_StartSDK() has not been called.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.

Connecting/Disconnecting Camera Devices

The following section describes the processes to actually connect and disconnect a camera device.

➢ PR_ConnectCamera

Connects a camera device. After this function is executed, actual communication with the camera will be established to enable various functions, such as remote release control and retrieval/setting of device properties.

Syntax

```
prCAPI PR_ConnectCamera(
     prHandle  CameraHandle
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle corresponding to the camera device to be connected.

Error ID

Error ID	Description
prINVALID_FN_CALL	CreateCameraObject() has not been called.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.
prMEM_ALLOC_FAILED	Insufficient memory error.
Other	Error value of GetLastError() in Win32 API, or STI/WIA interface error value.

Reference:

Note: Even within a process, multiple connections cannot be established with a single camera device using PR_ConnectCamera() by means of threads (operation cannot be guaranteed). The same applies when multiple camera handles have been created for a single camera device.

➢ PR_DisconnectCamera

Disconnects a camera device that has been connected by PR_ConnectCamera().
If communication with a camera device has become no longer necessary, be sure to

disconnect the camera using this function. If the camera is not disconnected, memory leaks or other problems may occur.

Syntax

```
prCAPI PR_DisconnectCamera(
     prHandle   CameraHandle
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle corresponding to the camera device to be disconnected.

Error ID

Error ID	Description
prINVALID_FN_CALL	CreateCameraObject() has not been called.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.
Other	Error value of GetLastError() in Win32 API, or STI/WIA interface error value.

Reference:

After this function is executed, the client will no longer be able to communicate with the camera device. To allow the client to communicate with the camera device again, connect the camera again using PR_ConnectCamera(). As long as the corresponding camera handle has not been deleted with PR_DestroyCameraObject(), the same camera handle can be used to connect the camera again.

Retrieving Camera Events

The following section describes the function used to retrieve events from a camera device. The client is notified of event information from each camera device by having the callback function specified in the client called by the PS-ReC SDK.

➢ PR_SetEventCallBack

The callback functions used for retrieving event information from camera devices are registered in the PS-ReC SDK. There is no need to execute this function if event information need not be retrieved from camera devices.

Syntax

```
prCAPI  PR_SetEventCallBack(
     prHandle       CameraHandle,
     prContext      Context,
     prSetEventCB*  pSetEventCB
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle corresponding to the camera device to register the callback function for.
Context	[in] Specifies the data to be passed to the parameters in the registered callback function. The client may use this freely.
pSetEventCB	[in] Specifies the pointer to the callback function to be registered.

Error ID

Error ID	Description
prINVALID_FN_CALL	CreateCameraObject() has not been called.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.
prINVALID_PARAMETER	The specified parameter is invalid.
prEVENT_CALLBACK_EXIST	A callback function is already registered.
Other	Error value of GetLastError() in Win32 API, or STI/WIA interface error value.

Reference:

Note 1: Only one callback function can be registered for retrieving events from each camera. Two or more callback functions cannot be registered at a time.

Note 2: A callback function can be registered without connecting the applicable camera device. However, event information from the camera can only be retrieved after the camera device has been connected (after PR_ConnectCamera() has been called).

Note 3: To transfer remotely captured images from the camera device to the computer, camera event information is required. For details, refer to "PR_GetReleasedData()."

For the type definition regarding the SetEventCB callback function, refer to Chapter 4, "Structure and Callback Function Definitions" or "prFuncType.h."

➢ PR_ClearEventCallBack

Releases the callback function for retrieving event information that has been registered by PR_SetEventCallBack(). If retrieval of event information from a camera device has become no longer necessary, be sure to release the applicable callback function using this function. If the callback function is not released, memory leaks or other problems may occur.

Syntax

```
prCAPI  PR_ClearEventCallBack(
     prHandle        CameraHandle
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle corresponding to the camera device to register the callback function for.

Error ID

Error ID	Description
prINVALID_FN_CALL	CreateCameraObject() has not been called.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.
Other	Error value of GetLastError() in Win32 API.

Reference:

After this function is executed, event information can no longer be retrieved from the applicable camera device even when an event occurs in the camera device while the camera device is connected. To retrieve event information from the camera device again, register a callback function for the camera device again using PR_SetEventCallback(). As long as the corresponding camera handle has not been deleted with PR_DestroyCameraObject(), the same camera handle can be used to register the callback function again.

Retrieving Camera Device Performance Information

The following describes the function used to retrieve information based on which to determine the performance of a connected camera.

➢ PR_GetDeviceInfo

Retrieves performance information regarding a connected camera. This information is called "DeviceInfo data set." The retrieved DeviceInfo data set is used to determine if the camera device supports remote capture, or which capture setting items (device properties) are valid.

Syntax

```
prCAPI  PR_GetDeviceInfo(
     prHandle        CameraHandle,
     prUInt32*       pBufferSize,
     prVoid*         pDeviceInfo
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle corresponding to the connected camera.
pBufferSize	[in/out] Specifies the buffer size of the next parameter. If the function has been executed successfully, the data size stored in the buffer is set.
pDeviceInfo	[out] Specifies the pointer indicating the buffer. The data size is stored in the data set type of DeviceInfo (variable length) described in Reference.

Error ID

Error ID	Description
prINVALID_FN_CALL	CreateCameraObject() has not been called.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.
prINVALID_PARAMETER	The specified parameter is invalid.
prMEM_ALLOC_FAILED	Insufficient memory error.
prINSUFFICIENT_BUFFER	The buffer size specified by the parameter is smaller than the actual data size. Or, the buffer pointer is NULL.
Other	Error value of GetLastError() in Win32 API.

Reference:

The data structure of DeviceInfo data set is described.

Data field	Data type	Description
StandardVersion	prUInt16	Standard version.
VendorExtensionID	prUInt32	Vendor extended ID.
VendorExtensionVersion	prUInt16	Vendor extended version.
VendorExtensionDesc	String (Refer to Attached Table 1)	Vendor extended information.
FunctionalMode	prUInt16	Function mode.
OperationsSupported	prptpOperationCode Array (Refer to Attached Table 2)	Array of operations supported by the camera device.
EventsSupported	prptpEventCode Array (Refer to Attached Table 3)	Array of events supported by the camera device.
DevicePropertiesSupported	prptpDevicePropCode Array (Refer to Attached Table 4)	Array of device properties supported by the camera device.
CaptureFormats	prptpObjectFormatCode Array (Refer to Attached Table 5)	Array of captured image types supported by the camera device.
ImageFormats	prptpObjectFormatCode Array (Refer to Attached Table 5)	Array of image types supported by the camera device.
Manufacturer	String(Refer to Attached Table 1)	Company information.
Model	String(Refer to Attached Table 1)	Model name.
DeviceVersion	String(Refer to Attached Table 1)	Device version.
SerialNumber	String(Refer to Attached Table 1)	Serial number.

Attached Table 1. String Type

Data field	Data type	Description
NumChars	prUInt8	Number of characters in StringChars.
StringChars	prWChar array	Unicode character string (consisting of up to 255 characters, ending with a NULL).

Attached Table 2. prptpOperationCode Array Type

Data field	Data type	Description
NumElements	prUInt32	Number of elements in OperationCode.
OperationCode	prptpOperationCode array	Operation codes supported by the camera device. Refer to "prType.h" for the retrievable values.

Attached Table 3. prptpEventCode Array Type

Data field	Data type	Description
NumElements	prUInt32	Number of elements in EventCode.
EventCode	prptpEventCode array	Event codes supported by the camera device. Refer to "prType.h" or Chapter 6, "Events" for the retrievable values.

Attached Table 4. prptpDevicePropCode Array Type

Data field	Data type	Description
NumElements	prUInt32	Number of elements in DevicePropertyCode.
DevicePropCode	prptpDevicePropCode array	Device property codes supported by the camera device. Refer to "prType.h" or Chapter 5, "Device Properties" for the retrievable values.

Attached Table 5. prptpObjectFormatCode Array Type

Data field	Data type	Description
NumElements	prUInt32	Number of elements in ObjectFormatCode.
ObjectFormatCode	prptpObjectFormatCode Array	Object type codes supported by the camera device. Refer to "prType.h" for the retrievable values.

Note: The DeviceInfo data set is variable length data, so the PS-ReC SDK does not provide a structure definition for DeviceInfo. For the procedure to analyze data, refer to Chapter 7, "Sample" or the samples provided in the PS-ReC SDK medium.

If the pDeviceInfo buffer size specified by the pBufSize parameter is smaller than the actual data size of the DeviceInfo data set to be retrieved, the actual data size will be set in pBufSize and prINSUFFICIENT_BUFFER will be returned. The same applies when pDeviceInfo is a NULL pointer. Conversely, if pBufSize is greater than the actual data size, all data will be stored in the buffer and the stored data size will be set in pBufSize.

Remote Release Control Functions

This section describes the functions used for remote release control. Remote release control refers to the ability to operate a camera connected to a computer remotely from the PS-ReC SDK, such as operating the camera shutter from the computer to capture still images.

Specifically, the following items are described:

- Basic Functions
- Viewfinder Function
- AF Lock Settings
- Notes on Using Remote Release Control Functions

Note 1: The remote release control functions can be executed only when the client is connected to a camera device.

Note 2: To access a camera device using the WIA interface, end the remote release control mode, and then access the camera device.
For details, refer to Chapter 7, "Sample."

Basic Functions

The following section describes the functions used to perform basic operations, such as starting the remote release control, capturing an image, and retrieving the captured image.

➢ PR_InitiateReleaseControl

Starts the remote release control mode.

To use remote release control, you must first execute this function to switch the camera device to the remote release control mode.

Syntax

```
prCAPI  PR_InitiateReleaseControl(
     prHandle       CameraHandle
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle corresponding to the connected camera device.

Error ID

Error ID	Description
prINVALID_FN_CALL	ConnectCamera() has not been called.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.
prMEM_ALLOC_FAILED	Insufficient memory error.
Other	Error value of GetLastError() in Win32 API, PTP error value, or STI/WIA interface error value.

Reference:

Note: All of the subsequent functions described in this section can only be executed in the remote release control mode.

➢ PR_TerminateReleaseControl

Exits the remote release control mode.

If you have switched the camera device to the remote release control mode by executing PR_InitiateReleaseControl, you must exit the remote release control mode using this function.

Syntax

```
ceCAPI  PR_TerminateReleaseControl(
     prHandle        CameraHandle
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle corresponding to the connected camera device.

Error ID

Error ID	Description
prINVALID_FN_CALL	The camera device is not in the remote release control mode.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.
prMEM_ALLOC_FAILED	Insufficient memory error.
Other	Error value of GetLastError() in Win32 API, PTP error value, or STI/WIA interface error value.

➢ PR_RC_Release

Captures an image (sends a capture command to the camera device).

Syntax

```
prCAPI  PR_RC_Release(
     prHandle        CameraHandle
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle corresponding to the connected camera device.

Error ID

Error ID	Description
prINVALID_FN_CALL	The camera device is not in the remote release control mode.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.
prMEM_ALLOC_FAILED	Insufficient memory error.
Other	Error value of GetLastError() in Win32 API, PTP error value, or STI/WIA interface error value.

➢ PR_RC_GetReleasedData

Retrieves the image data generated by the capture via PR_RC_Release().

Once event information has been retrieved from the camera device indicating the object handle corresponding to the image data and whether main image or thumbnail has been generated, this function is used to retrieve the image data, based on the judgment of the event information.

Syntax

```
prCAPI PR_RC_GetReleasedData(
     prHandle            CameraHandle,
     prObjectHandle      ObjectHandle,
     prptpEventCode      EventCode,
     prUInt32            TransSize,
     prContext           Context,
     prGetFileDataCB*    pGetFileDataCB
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle corresponding to the connected camera device.
ObjectHandle	[in] Specifies the object handle corresponding to the image data to be retrieved.
EventCode	[in] Specifies the event code indicating the type (main image or thumbnail) of the image data to be retrieved. The event code is determined from the event information retrieved from the camera device and specified. prPTP_FULL_VIEW_RELEASED: Retrieves main image data. prPTP_THUMBNAIL_RELEASED: Retrieves thumbnail image data. Also refer to Chapter 6, "Events."
TransSize	[in] Specifies the data size of a single transfer when image data is divided and transferred over multiple times.
Context	[in] Specifies the data to be passed to the parameters in the registered callback function. The client may use this freely.
pGetFileDataCB	[in] Specifies the pointer to the callback function to be registered.

Error ID

Error ID	Description
prINVALID_FN_CALL	The camera device is not in the remote release control mode.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.
prMEM_ALLOC_FAILED	Insufficient memory error.
prINVALID_PARAMETER	The specified parameter is invalid.
Other	Error value of GetLastError() in Win32 API, PTP error value, or STI/WIA interface error value.

Reference:

Once this function has been executed, the PS-ReC SDK stores the captured image data from the camera device in the allocated buffer in the PS-ReC SDK every time the TransSize parameter is received. Thereafter, the PS-ReC SDK calls the client's callback function that has been specified by the pGetFileDataCB parameter. At this time, the buffer address, size, offset, progress, and other information in the PS-ReC SDK are passed to the parameters in the callback function.

Based on the parameter information, the client should periodically copy the image data, within the callback function, to the buffer provided in the client or to a file.

After the client has finished processing the callback function and the function has been returned, the PS-ReC SDK frees the buffer and then repeats this process until all captured image data is retrieved.

The client can save the captured image data to a file by, for example, periodically appending to a desired file each data passed by the callback function.

For information about the prGetFileDataCB callback function, refer to Chapter 4, "Structure and Callback Function Definitions."

➢ PR_RC_GetNumAvailableShot

Retrieve the number of remaining image data that can be saved on the internal memory card of the camera device. This information indicates how many more shots you can take with the camera device.

Syntax

```
prCAPI PR_RC_GetNumAvailableShot(
     prHandle        CameraHandle,
     prUInt32*       pNum
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle corresponding to the connected camera device.
pNum	[out] Returns the number of remaining shots.

Error ID

Error ID	Description
prINVALID_FN_CALL	The camera device is not in the remote release control mode.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.
prMEM_ALLOC_FAILED	Insufficient memory error.
prINVALID_PARAMETER	The specified parameter is invalid.
Other	Error value of GetLastError() in Win32 API, PTP error value, or STI/WIA interface error value.

Viewfinder Function

The following describes the functions relating to the Viewfinder function in the remote release control mode.

The Viewfinder function retrieves live image data from the camera. By continuously retrieving and displaying the data, this function makes it possible to display in an application the same image that is in the camera finder.

Note: Not all camera models support the Viewfinder function.

➢ PR_RC_StartViewfinder

Starts the Viewfinder mode.

To use the Viewfinder function, you must execute this function to switch the camera device to the Viewfinder mode. While in the Viewfinder mode, the camera device generates image data for the Viewfinder.

Syntax

```
prCAPI PR_RC_StartViewFinder(
     prHandle            CameraHandle,
     prContext           Context,
     prViewFinderCB*     pViewFinderCB
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle corresponding to the connected camera device.
Context	[in] Specifies the data to be passed to the parameters in the registered callback function. The client may use this freely.
pViewFinderCB	[in] Specifies the pointer to the callback function to be registered.

Error ID

Error ID	Description
prINVALID_FN_CALL	The camera device is not in the remote release control mode.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.
prMEM_ALLOC_FAILED	Insufficient memory error.
prINVALID_PARAMETER	The specified parameter is invalid.
Other	Error value of GetLastError() in Win32 API, PTP error value, or STI/WIA interface error value.

Reference:

Once this function has been executed, the PS-ReC SDK retrieves Viewfinder data from the camera device, stores the data in the buffer allocated in the PS-ReC SDK, and then calls the client's callback function specified by the pViewFinderCB parameter. At this time, the buffer address, size, and other information in the PS-ReC SDK are passed to the parameters in the callback function.

Based on the parameter information, the client should display the image data within the callback function. The Viewfinder function is achieved by repeating this process.

After the client has finished processing the callback function and the function has been returned, the PS-ReC SDK frees the buffer, and then retrieves Viewfinder data from the camera and allocates the buffer again.

For information about the prViewFinderCB callback function, refer to Chapter 4, "Structure and Callback Function Definitions."

➢ PR_RC_TermViewfinder

Exits the Viewfinder mode.

This function must be executed to switch the camera device out of the Viewfinder mode after executing the PR_RC_StartViewfinder function.

Syntax

```
prCAPI PR_RC_TermViewFinder(
     prHandle       CameraHandle
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle corresponding to the connected camera device.

Error ID

Error ID	Description
prINVALID_FN_CALL	The camera device is not in the viewfinder mode.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.
prMEM_ALLOC_FAILED	Insufficient memory error.
Other	Error value of GetLastError() in Win32 API, PTP error value, or STI/WIA interface error value.

➢ PR_RC_DoAeAfAwb

Instructs the camera in the Viewfinder mode to reset the auto exposure (AE), auto focus (AF), or auto white balance (AWB). After this function is executed, the specified reset item will be reflected in the Viewfinder data.

Syntax

```
prCAPI  PR_RC_DoAeAfAwb(
     prHandle                  CameraHandle,
     prptpAeAfAwbResetFlag     ResetFlag
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle corresponding to the connected camera device.
ResetFlag	[in] Specifies the item to be reset. One of the following values is set: prptpAEAFAWB_RESET_AE: Reset AE prptpAEAFAWB_RESET_AF: Reset AF prptpAEAFAWB_RESET_AWB: Reset AWB

Error ID

Error ID	Description
prINVALID_FN_CALL	ConnectCamera() has not been called.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.
prMEM_ALLOC_FAILED	Insufficient memory error.
Other	Error value of GetLastError() in Win32 API, PTP error value, or STI/WIA interface error value.

Reference:

As long as this function is not executed, the exposure and focus settings for Viewfinder data remain fixed at the values that were set when the Viewfinder mode was started. As a result, if the distance to the subject or the brightness changes, you must execute this function to reset the AE, AF, and AWB.

AF Lock Settings

This section describes the functions relating to the AF lock and unlock settings on a camera device.

➢ PR_RC_FocusLock

Lock the AF.

After this function is executed, the camera will adjust the focus, and then enter the manual focus mode.

Syntax

```
prCAPI  PR_RC_FocusLock(
     prHandle   CameraHandle
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle corresponding to the connected camera device.

Error ID

Error ID	Description
prINVALID_FN_CALL	ConnectCamera() has not been called.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.
prMEM_ALLOC_FAILED	Insufficient memory error.
Other	Error value of GetLastError() in Win32 API, PTP error value, or STI/WIA interface error value.

➢ PR_RC_FocusUnlock

Unlock the AF.

Syntax

```
prCAPI  PR_RC_FocusUnlock(
     prHandle   CameraHandle
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle corresponding to the connected camera device.

Error ID

Error ID	Description
prINVALID_FN_CALL	ConnectCamera() has not been called.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.
prMEM_ALLOC_FAILED	Insufficient memory error.
Other	Error value of GetLastError() in Win32 API, PTP error value, or STI/WIA interface error value.

Reference:

After this function is executed and the AF is unlocked, the AF mode ("Normal," "Macro," "Snapshot," "Distant View," and the like) will switch to "Normal" regardless of the mode that was effective before the AF lock. To return the camera to the AF mode effective before the AF lock, you must set the AF mode again.

Notes on Using Remote Release Control Functions

Depending upon the camera model, all or some of the remote release control function, Viewfinder function, and AF lock setting function may be unavailable. Accordingly, some functions provided by the PS-ReC SDK cannot be used depending upon the camera model.

To determine if a given function can be used on a specific camera model, check "OperationSupported" in the device performance information (DeviceInfo data set) that can be retrieved using PR_GetDeviceInfo(), to see if the operation code supports the function.

For the procedure to determine the availability of operation codes, refer to Chapter 7, "Sample" or the samples provided in the PS-ReC SDK medium.

Refer to "prType.h" for the definition values of operation codes.

Basic Functions

PS-ReC SDK Functions	Definition Values of Operation Codes
PR_InitiateReleaseControl()	prPTP_INITIATE_RELEASE_CONTROL
PR_TerminateReleaseControl()	prPTP_TERMINATE_RELEASE_CONTROL
PR_RC_Release()	prPTP_RC_CAPTURE

Note: You can determine whether a given camera device supports the remote release control function by checking if prPTP_INITIATE_RELEASE_CONTROL is available.

Viewfinder Function

PS-ReC SDK Functions	Definition Values of Operation Codes
PR_RC_StartViewFinder()	prPTP_RC_INITIATE_VIEW_FINDER
PR_RC_TermViewFinder()	prPTP_RC_TERMINATE_VIEW_FINDER
PR_RC_DoAeAfAwb()	prPTP_RC_RELEASE_DO_AE_AF_AWB

Note: You can determine whether a given camera device supports the Viewfinder function by checking if prPTP_RC_INITIATE_VIEW_FINDER is available.

AF Lock Settings

PS-ReC SDK Functions	Definition Values of Operation Codes
PR_RC_FocusLock()	prPTP_RC_FOCUS_LOCK
PR_RC_FocusUnlock()	prPTP_RC_FOCUS_UNLOCK

Note: You can determine whether a given camera device supports the AF lock function by checking if prPTP_RC_FOCUS_LOCK is available.

Device Property Functions

This function describes the device property functions.

Device properties are data stored in a camera device to indicate the condition, capacity, settings, and other information of the camera. The device property functions are used to enumerate the settable property values of each device or to reference or change the current settings.

For details on the device properties supported by the PS-ReC SDK, refer to Chapter 5, "Device Properties."

Note 1: The device property functions can be executed only when the client is connected to (communicating with) a camera device.

Note 2: The types of properties that are supported may differ depending upon the camera model. The types of supported properties may also differ depending upon whether or not the camera is in the remote release control mode.

➢ PR_GetDevicePropDesc

Retrieves information indicating the data type of each device property, whether or not the property can be referenced/set, the factory setting, the current setting, and the enumeration and range of settable values. The data that can be retrieved with PR_GetDevicePropDesc() are called "DevicePropDesc data set."

Syntax

```
prCAPI  PR_GetDevicePropDesc(
     prHandle                  CameraHandle,
     prptpDevicePropCode       DevicePropCode,
     prUInt32*                 pBufferSize,
     prVoid*                   pDevicePropDesc
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle corresponding to the connected camera device.
DevicePropCode	[in] Specifies the device property code to retrieve the DevicePropDesc data set for. For the device properties you can specify, refer to Chapter 5, "Device Properties."
pBufferSize	[in/out] Specifies the buffer size of the next parameter. If the function has been executed successfully, the data size stored in the buffer is set.
pDevicePropDesc	[out] Specifies the pointer indicating the buffer. The data size is stored in the data set type of DevicePropDesc (variable length) described in "Reference".

Error ID

Error ID	Description
prINVALID_FN_CALL	CreateCameraObject() has not been called.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.
prINVALID_PARAMETER	The specified parameter is invalid.
prMEM_ALLOC_FAILED	Insufficient memory error.
prINSUFFICIENT_BUFFER	The buffer size specified by the parameter is smaller than the actual data size. Or, the buffer pointer is NULL.
Other	Error value of GetLastError() in Win32 API, PTP error value, or STI/WIA interface error value.

Reference:

The data structure of DevicePropDesc data set is described.

Data field	Data type	Description
Device Property Code	prptpDevicePropCode	Device Property Codes Refer to Chapter 5, "Device Properties."
DataType	prUInt16	Code indicating the data type of the device property (Refer to Attached Table 1)
GetSet	prUInt8	Flag indicating whether the device property is Read-Only (Get) or Read-Write (Get/Set) (Refer to Attached Table 2)
Factory Default Value	(Data type indicated in DataType)	Factory setting of the device property
Current Value	(Data type indicated in DataType)	Current setting of the device property
Form Flag	prUInt8	Flag indicating the format of the next FORM field (Refer to Attached Table 3)
FORM	(Variable length)	Settable values of the device property (Refer to Attached Tables 4 and 5)

Attached Table 1. List of DataType Codes

DataType code	Data type indicated by the code	Description
0x0000	(Not defined)	Not defined
0x0001	prInt8	Signed 8 bit integer
0x0002	prUInt8	Unsigned 8 bit integer
0x0003	prInt16	Signed 16 bit integer
0x0004	prUInt16	Unsigned 16 bit integer
0x0005	prInt32	Signed 32 bit integer
0x0006	prUInt32	Unsigned 32 bit integer
0x0007	prInt64	Signed 64 bit integer
0x0008	prUInt64	Unsigned 64 bit integer
0x0009	(Not defined)	Signed 128 bit integer
0x000A	(Not defined)	Unsigned 128 bit integer
0x4001	prInt8 array	Array of Signed 8 bit integer
0x4002	prUInt8 array	Array of Unsigned 8 bit integer
0x4003	prInt16 array	Array of Signed 16 bit integer
0x4004	prUInt16 array	Array of Unsigned 16 bit integer
0x4005	prInt32 array	Array of Signed 32 bit integer
0x4006	prUInt32 array	Array of Unsigned 32 bit integer
0x4007	prInt64 array	Array of Signed 64 bit integer
0x4008	prUInt64 array	Array of Unsigned 64 bit integer
0x4009	(Not defined)	Array of Signed 128 bit integer
0x400A	(Not defined)	Array of Unsigned 128 bit integer
0xFFFF	String(Refer to Attached Table 1-2)	Unicode character string of variable length
Other value	(Not defined)	(Reserved)

Attached Table 1-2. String Type

Data field	Data type	Description
NumChars	prUInt8	Number of characters in StringChars.
StringChars	prWChar array	Unicode character string (consisting of up to 255 characters, ending with a NULL).

Attached Table 2. Description of Get/Set Flag Values

Flag value	Description
0x00	Indicates Get(Read-Only)
0x01	Indicates Get/Set(Read-Write)

Attached Table 3. Description of Form Flag Values

Flag value	Description
0x00	There is no FORM field.
0x01	The FORM field is of Range-Form type (refer to Table 4).
0x02	The FORM field is of Enumeration-Form type (refer to Table 5).

Attached Table 4. Range-Form Type (Range of Settable Values)

Data field	Data type	Description
Minimum Value	(Data type indicated in DataType of DevicePropDesc)	Minimum value supported by the device property
Maximum Value	(Data type indicated in DataType of DevicePropDesc)	Maximum value supported by the device property
Step Size	(Data type indicated in DataType of DevicePropDesc)	Step from the minimum value to maximum value of the device property

Attached Table 5. Enumeration-Form Type (Array of Enumerated Settable Values)

Data field	Data type	Description
NumElements	prUInt16	Number of elements in SupportedValue.
SupportedValue	(Array of data type indicated in DataType of DevicePropDesc)	Array of settable values supported by the device property

Note: The DevicePropDesc data set is variable length data, so the PS-ReC SDK does not provide a structure definition or definition value for DevicePropDesc. For the procedure to analyze data, refer to Chapter 7, "Sample" or the samples provided in the PS-ReC SDK medium.

If the pDevicePropDesc buffer size specified by the pBufSize parameter is smaller than the actual data size of the DevicePropDesc data set to be retrieved, the actual data size will be set in pBufSize and prINSUFFICIENT_BUFFER will be returned. The same applies when pDevicePropDesc is a NULL pointer. Conversely, if pBufSize is greater than the actual data size, all data will be stored in the buffer and the stored data size will be set in pBufSize.

➢ PR_GetDevicePropValue

Retrieves the current setting of each device property.

Syntax

```
prCAPI  PR_GetDevicePropValue(
     prHandle                  CameraHandle,
     prptpDevicePropCode       DevicePropCode,
     prUInt32*                 pBufferSize,
     prVoid*                   pDeviceProperty
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle corresponding to the connected camera device.
DevicePropCode	[in] Specifies the device property code. For the device property codes you can specify, refer to Chapter 5, "Device Properties."
pBufferSize	[in/out] Specifies the buffer size of the next parameter. If the function has been executed successfully, the data size stored in the buffer is set.
pDeviceProperty	[out] Specifies the pointer indicating the buffer. Stores the current setting of the device property.

Error ID

Error ID	Description
prINVALID_FN_CALL	CreateCameraObject() has not been called.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.
prINVALID_PARAMETER	The specified parameter is invalid.
prMEM_ALLOC_FAILED	Insufficient memory error.
prINSUFFICIENT_BUFFER	The buffer size specified by the parameter is smaller than the actual data size. Or, the buffer pointer is NULL.
Other	Error value of GetLastError() in Win32 API, PTP error value, or STI/WIA interface error value.

Reference:

If the pDevicePropDesc buffer size specified by the pBufSize parameter is smaller than the actual data size of the DevicePropDesc data set to be retrieved, the actual data size will be set in pBufSize and prINSUFFICIENT_BUFFER will be returned. The same applies when pDevicePropDesc is a NULL pointer. Conversely, if pBufSize is greater than the actual data size, all data will be stored in the buffer and the stored data size will be set in pBufSize.

➢ PR_SetDevicePropValue

Changes the current setting of each device property to a specified value.

Syntax

```
prCAPI  PR_SetDevicePropValue(
     prHandle                  CameraHandle,
     prptpDevicePropCode       DevicePropCode,
     prUInt32                  DataSize,
     prVoid*                   pDeviceProperty
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle corresponding to the connected camera device.
DevicePropCode	[in] Specifies the device property code. For the device property codes you can specify, refer to Chapter 5, "Device Properties."
DataSize	[in] Specifies the data size of the next parameter.
pDeviceProperty	[in] Specifies the pointer indicating the device property data to be set.

Error ID

Error ID	Description
prINVALID_FN_CALL	CreateCameraObject() has not been called.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.
prINVALID_PARAMETER	The specified parameter is invalid.
prMEM_ALLOC_FAILED	Insufficient memory error.
Other	Error value of GetLastError() in Win32 API, PTP error value, or STI/WIA interface error value.

Reference:

Note:	Some device properties cannot be changed. You can check if a given device property can be changed, by referencing the GetSet flag in the DevicePropDesc data set that can be retrieved using PR_GetDevicePropDesc().

➢ PR_RC_GetChangedReleaseParamesList

Retrieves a list of device property codes relating to capture parameters, which can be set on the applicable camera device or which have been changed.

On some camera devices, changing any device property relating to a capture parameter or zoom position may change other capture parameters (device properties). This function can produce an enumerated list of device property codes that must be retrieved again after the camera condition has changed as a result of parameter/property changes as specified above.

If any device property relating to a capture parameter has been changed, the client must always use this function to retrieve again the settable values and current setting of the applicable device property by using PR_GetDevicePropDesc() and/or PR_GetDevicePropValue().

Syntax

```
prCAPI  PR_RC_GetChangedReleaseParamesList(
     prHandle        CameraHandle,
     prUInt32*       pBufferSize,
     prVoid*         pParamsList
);
```

Parameter

Parameter	Description
CameraHandle	[in] Specifies the camera handle corresponding to the connected camera device.
pBufferSize	[in/out] Specifies the buffer size of the next parameter. If the function has been executed successfully, the data size stored in the buffer is set.
pParamsList	[out] Specifies the pointer indicating the buffer. A list of device properties is stored.

Error ID

Error ID	Description
prINVALID_FN_CALL	CreateCameraObject() has not been called.
prINVALID_HANDLE	The camera handle specified by the parameter is invalid.
prINVALID_PARAMETER	The specified parameter is invalid.
prMEM_ALLOC_FAILED	Insufficient memory error.
prINSUFFICIENT_BUFFER	The buffer size specified by the parameter is smaller than the actual data size. Or, the buffer pointer is NULL.
Other	Error value of GetLastError() in Win32 API, PTP error value, or STI/WIA interface error value.

Reference:

The data structure of the pParamsList parameter is described.

Data field	Data type	Description
NumElements	prUInt16	Number of elements in DevicePropCode.
DevicePropCode	prptpDevicePropCode array	Changed device property code.

Note 1: The PS-ReC SDK does not provide a structure that defines the pParamsList data structure. For the procedure to analyze data, refer to Chapter 7, "Sample" or the samples provided in the PS-ReC SDK medium.

Note 2: PR_RC_GetChangedReleaseParamesList() can be used only in the remote release control mode. Just like the remote release control functions, this function can be used as long as the operation code prPTP_RC_GET_CHANGED_RELEASE_PARAMS_LIST is available in "OperationSupported" in the DeviceInfo data set obtained by PR_GetDeviceInfo().

4 Structure and Callback Function Definitions

This chapter describes the structures and callback functions used in the PS-ReC SDK.

Structures

➢ prDllsVerInfo , prVerInfo

File version information structure. This structure is used with PR_GetDllsVersion().

The prDllsVerInfo structure consists of a prVerInfo structure array.

```
typedef struct{
     prWChar ModuleName[512];
     prWChar Version[32];
}prVerInfo;
```

Member	Description
ModuleName	Full path to the module name.
Version	File version number corresponding to ModuleName.

```
typedef struct{
     prUInt32  Entry;
     prVerInfo VerInfo[prANY];
}prDllsVerInfo;
```

Member	Description
Entry	Number of modules (number of VerInfo arrays) included in this structure.
VerInfo	Array of file version number information of PS-ReC SDK modules.

➢ prDeviceList , prDeviceInfoTable

Camera device information structure. This structure is used with PR_GetDeviceList().

The prDeviceList structure consists of a prDeviceInfoTable structure array.

```
typedef struct{
     prWChar         DeviceInternalName[512];
     prWChar         ModelName[32];
     prUInt16        Generation;
     prUInt32        Reserved1;
     prUInt32        ModelID;
     prUInt16        Reserved2;
     prPorttype      PortType;
     prUInt32        Reserved3;
}prDeviceInfoTable;
```

Member	Description
DeviceInternalName	Internal device name.
ModelName	Camera model name.
Generation	Camera generation number (only the upper 8 bits are effective).
Reserved1	Reserved. This should be ignored.
ModelID	Camera model ID.
Reserved2	Reserved. This should be ignored.
PortType	Indicates the port at which the camera is connected to the computer. prPORTTYPE_WIA: Connected via WIA. prPORTTYPE_STI: Connected via STI.
Reserved3	Reserved. This should be ignored.

Note 1: DeviceInternalName (internal device name) is the same as the internal device name used when the camera is connected via the STI or WIA interface. In other words, each camera device can be connected via STI/WIA using DeviceInternalName. For details, refer to Chapter 7, "Sample."

Note 2: To retrieve generation number information using Generation, use the prSUB_GENERATION_CAMERA(gen) macro. For the macros, refer to "prType.h." Ignore the lower 8 bits of Generation.

```
typedef struct{
     prUInt32             NumList;
     prDeviceInfoTable    DeviceInfo[prANY];
}prDeviceList;
```

Member	Description
NumList	Number of camera device information (number of DeviceInfo arrays) included in this structure.
DeviceInfo	Camera device information array.

➢ prProgress

Structure indicating the progress of transfer of captured image data. It is used with the PR_GetReleasedDataceGetFileDataCB() callback function.

```
typedef struct{
     prProgressMsg   lMessage;
     prProgressSts   lStatus;
     prUInt32        lPercentComplete;
     prUInt32        lOffset;
     prUInt32        lLength;
     prUInt32        lReserved;
     prUInt32        lResLength;
     prUInt8*        pbData;
}prProgress;
```

Member	Description
lMessage	Message indicating the reason for callback. Refer to Attached Table 1.
lStatus	Reserved. This should be ignored.
lPercentComplete	Indicates the current progress of transfer based on 100% representing the entire captured image data. The unit is percent.
lOffset	Indicates the offset of the transferred data pbData from the beginning of the entire captured image data. The unit is bytes.
lLength	Size of the transferred data pbData. The unit is bytes.
lReserved	Reserved. This should be ignored.
lResLength	Reserved. This should be ignored.
pbData	Pointer to the buffer in which the transferred data is stored. Captured image data is divided and stored separately.

Attached Table 1. Definition Values of prProgressMsg

Definition value	Description
MSG_DATA_HEADER	Message indicating the start of image data. When this message is displayed, the transferred data is not passed to a callback function. pbData is a NULL pointer.
MSG_DATA	Message indicating transfer of image data. When this message is displayed, the transferred data is passed to a callback function as necessary.
MSG_TERMINATION	Message indicating the end of image data. When this message is displayed, the transferred data is not passed to a callback function. pbData is a NULL pointer.

Callback Functions

This section describes the callback functions used in the PS-ReC SDK. The callback functions are implemented by the client and registered in the PS-ReC SDK when retrieving event information from a camera device or information regarding the progress of the transfer of captured image data. The PS-ReC SDK calls the registered callback functions as required.

➢ prSetEventCB

This function is called when a camera device event occurs. At the same time, the event information data is passed to the parameters of this callback function.

To retrieve camera device event information, the client must implement this function and register it in the PS-ReC SDK using PR_SetEventCallBack(). For information about PR_SetEventCallBack(), refer to Chapter 3, "Function References."

Syntax

```
typedef prResponse prSTDCALL  prSetEventCB(
     prHandle       CameraHandle,
     prContext      Context,
     prVoid*        pEventData
);
```

Parameter

Parameter	Description
CameraHandle	Camera handle corresponding to the camera device that generated the event.
Context	Context handle. The data specified in the Context parameter of PR_SetEventCallBack() is passed directly.
pEventData	Pointer to the event information data.

Return Value

Normally, prOK is returned.

Reference:

The structure of event information data is described.

Data field	Data type	Description
InterruptDataLength	prUInt32	Length of entire event data. The unit is bytes.
ContainerType	prUInt16	Reserved. This should be ignored.
EventCode	prUInt16	Event code.
TransactionID	prUInt32	Reserved. This should be ignored.
Parameter 1	prUInt32	Parameter data 1 accompanying the event.
Parameter 2	prUInt32	Parameter data 2 accompanying the event.
• • •	• • •	• • •
Parameter N	prUInt32	Parameter data N accompanying the event.

Use the formula below to calculate the number of parameters 1 to N, as the specific number varies depending on the event code:

Number of parameters N = (InterruptDataLength - 12) / 4

For the types and details of event codes, refer to Chapter 6, "Events."

Note: The PS-ReC SDK does not provide a structure that defines the event information data structure. For the procedure to analyze data, refer to Chapter 7, "Sample" or the samples provided in the PS-ReC SDK medium.

➢ prGetFileDataCB

This function is called periodically while captured image data is transferred. At the same time, data and progress information are passed to the parameters of this callback function.

To retrieve captured image data, the client must implement this function and register it in the PS-ReC SDK using PR_RC_GetReleasedData(). For information about PR_RC_GetReleasedData(), refer to Chapter 3, "Function References."

The frequency at which this callback function is called varies depending upon the value set in the TransSize parameter (size of unit data transferred) of PR_RC_GetReleasedData().

Syntax

```
typedef prResponse prSTDCALL  prGetFileDataCB(
     prHandle        CameraHandle,
     prObjectHandle  ObjectHandle,
     prContext       Context,
     prProgress*     pProgress
);
```

Parameter

Parameter	Description
CameraHandle	Camera handle corresponding to the connected camera device.
ObjectHandle	Object handle corresponding to the image data to be retrieved.
Context	Context handle. The data specified in the Context parameter of PR_RC_GetReleasedData() is passed directly.

Return Value

pProgress	Pointer to the structure that indicates progress. Refer to the structure definition prProgress, or "prType.h."

To stop (cancel) the process, the client must return prOPERATION_CANCELLED as a return value for this function. Otherwise, prOK should be returned.

➢ prViewFinderCB

This function is called when Viewfinder data is retrieved from a camera device. At the same time, the Viewfinder data is passed to the parameters of the callback function.

To retrieve Viewfinder data, the client must implement this function and register it in the PS-ReC SDK using PR_RC_StartViewFinder(). The PS-ReC SDK calls the client callback function once for each frame of Viewfinder data received. If the Viewfinder data transferred by the parameter is displayed each time the callback function is called, the client can continuously display live images from the camera device.

Syntax

```
typedef prResponse prSTDCALL  prViewFinderCB (
     prHandle   CameraHandle,
     prContext  Context,
     prUInt32   Size,
     prVoid*    pVFData
);
```

Parameter

Parameter	Description
CameraHandle	Camera handle corresponding to the connected camera device.
Context	Context handle. The data specified in the Context parameter of PR_RC_StartViewFinder() is passed directly.
Size	Size of the Viewfinder data specified by the next pVFData parameter.
pVFData	Pointer to the buffer in which one frame of Viewfinder data is stored. This buffer is allocated by the PS-ReC SDK. The buffer is cleared when the callback function is returned.

Return Value

Return prOK.

5 Device Properties

This chapter describes the device properties supported by the PS-ReC SDK.

Device properties are data stored in a camera device to indicate the condition, capacity, settings, and other information of the camera. By using the functions relating to device properties as described in Chapter 3, the settable values of each device property can be enumerated or the current setting of a given device property can be referenced or changed.

Note 1: The types of device properties that are supported may differ depending upon the camera model. Before using the device property functions, retrieve the DeviceInfo data set using PR_GetDeviceInfo(). Device property codes that can be used will be enumerated in the DevicePropertiesSupported field.

Note 2: The device properties supported by a camera may differ depending upon whether or not the camera is in the remote release control mode. After PR_InitiateReleaseControl() has been called, retrieve the DeviceInfo data set again using PR_GetDeviceInfo() to check the device property codes that can be used.

Note 3: The supported values for the settable/retrievable values of each device property described in this chapter may differ depending upon the camera model and other capture settings. In other words, not all values are always supported. Use PR_GetDevicePropDesc() to check the settable/retrievable values of each device property.

Device properties are described in detail based on the structure of DevicePropDesc data set. For the DevicePropDesc data set, refer to PR_GetDevicePropDesc() in Chapter 3, "Function References."

➢ prPTP_DEV_PROP_BUZZER

Sets on/off of the device buzzer.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD001
DataType	prUInt8
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Value	Description
0x0000	On
0x0001	Off

➢ prPTP_DEV_PROP_BATTERY_KIND

Indicates the type of the battery installed in the device.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD002
DataType	prUInt16
DescForms	Enum
Get/Set	Get

Reference:

Indicates the retrievable values.

Value	Description
0x0000	Unknown
0x0001	AC power supply
0x0002	Lithium ion battery
0x0003	Nickel hydride battery
0x0004	Nickel cadmium battery
0x0005	Alkaline manganese battery

➢ prPTP_DEV_PROP_BATTERY_STATUS

Indicates the battery condition in the device.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD003
DataType	prUInt32
DescForms	Enum
Get/Set	Get

Reference:

Indicates the retrievable values.

Value	Description
0x00000000	Not defined.
0x00000001	NORMAL
0x00000002	WARNING_LV1
0x00000003	EMERGENCY
0x00000004	WARNING_LV0

➢ prPTP_DEV_PROP_COMP_QUALITY

Indicates the image quality set by a capture parameter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD006
DataType	prUInt8
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Value	Description
0x00	Not defined.
0x01	Economy
0x02	Normal
0x03	Fine
0x04	Lossless
0z05	SuperFine

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_FULLVIEW_FILE_FORMAT

Indicates the image type set by a capture parameter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD007
DataType	prUInt8
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Value	Description
0x00	Not defined.
0x01	JPEG
0x02	CRW

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_IMAGE_SIZE

Indicates the image size set by a capture parameter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD008
DataType	prUInt8
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Value	Description
0x00	Large
0x01	Medium 1
0x02	Small
0x03	Medium 2
0x07	Medium 3

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_SELFTIMER

Indicates the self-timer setting.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD009
DataType	prUInt16
DescForms	0 (None)
Get/Set	Get

Reference:

Indicates the retrievable values. The unit is 100 milliseconds. Currently, only the following three settings are supported.

Value	Description
0	Self-timer Not Used
0x0064	Self-timer: 10 Seconds
0x0014	Self-timer: 2 Seconds
0xFFFF	Invalid or the setting is not changed.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_STROBE_SETTING

Indicates the flash set by a capture parameter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD00A
DataType	prUInt8
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Value	Description
0x00	Off
0x01	Auto
0x02	On
0x03	Red Eye Suppression
0x04	Low-speed Synchronization
0x05	Auto + Red Eye Suppression
0x06	On + Red Eye Suppression

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_BEEP

Indicates the buzzer set by a capture parameter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD00B
DataType	prUInt8
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Value	Description
0x00	Off
0x01	On

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_EXPOSURE_MODE

Indicates the exposure mode set by a capture parameter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD00C
DataType	prUInt8
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Value	Description
0x00	Auto
0x01	P
0x02	Tv
0x03	Av
0x04	M
0x05	A_DEP
0x06	M_DEP
0x07	Bulb
0x80	CAMERAM
0x81	MYCOLOR
0x82	PORTRAIT
0x83	LANDSCAPE
0x84	NIGHTSCENE
0x85	FOREST
0x86	SNOW
0x87	BEACH
0x88	FIREWORKS
0x89	PARTY
0x8A	NIGHTSNAP
0x8B	STITCH
0x8C	MOVIE
0x8D	CUSTOM
0x8E	INTERVAL
0x8F	DIGITALMACRO
0x90	LONGSHUTTER
0x91	UNDERWATER
0x92	KIDSANDPETS
0x93	FASTSHUTTER
0x94	SLOWSHUTTER
0x95	CUSTOM1
0x96	CUSTOM2
0x97	NEUTRAL
0x98	GRAY
0x99	SEPIA
0x9A	VIVID
0x9B	SPORTS
0x9C	MACRO
0x9D	SUPERMACRO
0x9E	PANFOCUS
0x9F	BW
0xA0	FLASHINHIBIT

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_IMAGE_MODE

Indicates the image mode to be applied at capture.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD00D
DataType	prUInt8
DescForms	0 (None)
Get/Set	Get

Reference:

Indicates the retrievable values.

Value	Description
0x00	Auto
0x01	Manual
0x02	Distant View
0x03	High-speed Shutter
0x04	Low-speed Shutter
0x05	Night View
0x06	Grayscale
0x07	Sepia
0x08	Portrait
0x09	Sports
0x0A	Macro
0x0B	Monochrome
0x0C	Pan Focus
0x0D	Neutral
0x0E	Soft

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_DRIVE_MODE

Sets the drive mode.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD00E
DataType	prUInt8
DescForms	0 (None)
Get/Set	Get

Reference:

Indicates the retrievable values.

Value	Description
0x00	Single-frame Shooting
0x01	Continuous Shooting
0x02	Timer (Single) Shooting
0x04	Continuous Low-speed Shooting
0x05	Continuous High-speed Shooting
0xFF	Invalid or the setting is not changed.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_EZOOM

Indicates the starting position of electronic zoom set by a capture parameter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD00F
DataType	prUInt8
DescForms	0 (None)
Get/Set	Get

Reference:

Indicates the retrievable values.

Value	Description
0x00	Off
0x01	x2
0x02	x4
0x03	Smooth

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_ML_WEI_MODE

Set the metering method.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD010
DataType	prUInt8
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Value	Description
0x00	Center-weighted Metering
0x01	Spot Metering
0x02	Average Metering
0x03	Evaluative Metering
0x04	Partial Metering
0x05	Center-weighted Average Metering
0x06	Spot Metering Interlocked with AF Frame
0x07	Multi-Spot Metering
0xFF	Invalid or the setting is not changed.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_AF_DISTANCE

Indicates information relating to the search range in the AF mode.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD011
DataType	prUInt8
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Value	Description
0x0	Manual
0x01	Auto
0x02	Unknown
0x03	Zone Focus (Close-up)
0x04	Zone Focus (Very Close)
0x05	Zone Focus (Close)
0x06	Zone Focus (Medium)
0x07	Zone Focus (Far)
0x08	Zone Focus (Reserved 1)
0x09	Zone Focus (Reserved 2)
0x0A	Zone Focus (Reserved 3)
0x0B	Zone Focus (Reserved 4)
0xFF	Invalid or the setting is not changed.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_FOCUS_POINT_SETTING

Set the selection mode for focusing point.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD012
DataType	prUInt16
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Value	Description
0x0000	Invalid or the setting is not changed.
0x1000	Focusing Point on Center Only, Manual
0x1001	Focusing Point on Center Only, Auto
0x3000	Multiple Focusing Points (No Specification), Manual
0x3001	Multiple Focusing Points, Auto
0x3002	Multiple Focusing Points (Right)
0x3003	Multiple Focusing Points (Center)
0x3004	Multiple Focusing Points (Left)

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_WB_SETTING

Indicates the white balance set by a capture parameter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD013
DataType	prUInt8
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Value	Description
0	Auto
1	Daylight
2	Cloudy
3	Tungsten
4	Fluorescent
6	Preset
7	Fluorescent H
9	Color Temperature
10	Custom White Balance PC-1
11	Custom White Balance PC-2
12	Custom White Balance PC-3
13	Missing number
14	Fluorescent H
0xff	Invalid or the setting is not changed.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_SLOW_SHUTTER_SETTING

Sets the low-speed shutter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD014
DataType	prUInt8
DescForms	0 (None)
Get/Set	Get

Reference:

Indicates the retrievable values.

Value	Description
0x00	Off
0x01	Night View
0x02	On
0x03	Low-speed shutter function not available.
0xFF	Invalid or the setting is not changed.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_AF_MODE

Indicates the auto-focus mode set by a capture parameter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD015
DataType	prUInt8
DescForms	0 (None)
Get/Set	Get

Reference:

Indicates the retrievable values.

Value	Description
0x00	Single Shot
0x01	AI Servo
0x02	AI Focus
0x03	Manual
0x04	Continuous

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_IMAGE_STABILIZATION

Indicates the image stabilization processing.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD016
DataType	prUInt8
DescForms	0 (None)
Get/Set	Get

Reference:

Indicates the retrievable values.

Value	Description
0	Off
1	On
0xFF	Invalid or the setting is not changed.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_CONTRAST

Indicates the contrast set by a capture parameter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD017
DataType	prInt8
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Value	Description
-2	Low 2
-1	Low
0	Standard
1	High
2	High 2

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_COLOR_GAIN

Indicates the color compensation set by a capture parameter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD018
DataType	prInt8
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Value	Description
-2	Low 2
-1	Low
0	Standard
1	High
2	High 2

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_SHARPNESS

Indicates the sharpness set by a capture parameter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD019
DataType	prInt8
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Value	Description
-2	Low 2
-1	Low
0	Standard
1	High
2	High 2

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_SENSITIVITY

Sets the sensitivity.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD01A
DataType	prInt8
DescForms	0 (None)
Get/Set	Get

Reference:

Indicates the retrievable values.

Value	Description
0	Standard
1	Upper 1
2	Upper 2
0xFF	Invalid or not set.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_PARAMETER_SET

Sets the selectable mode of development parameters.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD01B
DataType	prUInt16
DescForms	0 (None)
Get/Set	Get

Reference:

Indicates the retrievable values.

Value	Description
0x0008	Standard Development Parameters
0x0010	Development Parameters 1
0x0020	Development Parameters 2
0x0040	Development Parameters 3
0xffff	Invalid or the setting is not changed.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_ISO

Indicates the ISO value set by a capture parameter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD01C
DataType	prUInt16
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

ISO value	prPTP_DEV_PROP_ISO value
Auto	0x00
6	0x28
12	0x30
25	0x38
50	0x40
64	0x43
100	0x48
200	0x50
400	0x58
800	0x60
1600	0x68
3200	0x70
6400	0x78

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_AV

Indicates the Av value set by a capture parameter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD01D
DataType	prUInt16
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Av value	prPTP_DEV_PROP_AV value
1.0	0x08
1.1	0x0b
1.2	0x0c
1.2 (1/3)	0x0d
1.4	0x10
1.6	0x13
1.8	0x14
1.8 (1/3)	0x15
2.0	0x18
2.2	0x1b
2.5	0x1c
2.5 (1/3)	0x1d
2.8	0x20
3.2	0x23
3.5	0x24
3.5 (1/3)	0x25
4.0	0x28
4.5 (1/3)	0x2b
4.5	0x2c
5.6 (1/3)	0x2d
5.6	0x30
6.3	0x33
6.7	0x34
7.1	0x35
8.0	0x38
9.0	0x3b
9.5	0x3c

Av value	prPTP_DEV_PROP_AV value
10	0x3d
11	0x40
13 (1/3)	0x43
13	0x44
14	0x45
16	0x48
18	0x4b
19	0x4c
20	0x4d
22	0x50
25	0x53
27	0x54
29	0x55
32	0x58
36	0x5b
38	0x5c
40	0x5d
45	0x60
51	0x63
54	0x64
57	0x65
64	0x68
72	0x6b
76	0x6c
81	0x6d
91	0x70

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_TV

Indicates the Tv value set by a capture parameter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD01E
DataType	prUInt16
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Tv value	prPTP_DEV_PROP_TV value
Bulb	0x04
30"	0x10
25"	0x13
20"	0x14
20" (1/3)	0x15
15"	0x18
13"	0x1b
10"	0x1c
10" (1/3)	0x1d
8"	0x20
6" (1/3)	0x23
6"	0x24
5"	0x25
4"	0x28
3"2	0x2b
3"	0x2c
2"5	0x2d
2"	0x30
1"6	0x33
1"5	0x34
1"3	0x35
1"	0x38
0"8	0x3b
0"7	0x3c
0"6	0x3d
0"5	0x40
0"4	0x43
0"3	0x44
0"3 (1/3)	0x45
1/4	0x48
1/5	0x4b
1/6	0x4c
1/6 (1/3)	0x4d
1/8	0x50
1/10 (1/3)	0x53
1/10	0x54
1/13	0x55
1/15	0x58
1/20 (1/3)	0x5b

Tv value	prPTP_DEV_PROP_TV value
1/20	0x5c
1/25	0x5d
1/30	0x60
1/40	0x63
1/45	0x64
1/50	0x65
1/60	0x68
1/80	0x6b
1/90	0x6c
1/100	0x6d
1/125	0x70
1/160	0x73
1/180	0x74
1/200	0x75
1/250	0x78
1/320	0x7b
1/350	0x7c
1/400	0x7d
1/500	0x80
1/640	0x83
1/750	0x84
1/800	0x85
1/1000	0x88
1/1250	0x8b
1/1500	0x8c
1/1600	0x8d
1/2000	0x90
1/2500	0x93
1/3000	0x94
1/3200	0x95
1/4000	0x98

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_EXPOSURE_COMP

Indicates the exposure compensation value.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD01F
DataType	prUInt8
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable values.

Exposure compensation value	prPTP_DEV_PROP_EXPOSURE_COMP value
+3	0x00
+2 (2/3)	0x03
+2 (1/2)	0x04
+2 (1/3)	0x05
+2	0x08
+1 (2/3)	0x0b
+1 (1/2)	0x0c
+1 (1/3)	0x0d
+1	0x10
+2/3	0x13
+1/2	0x14
+1/3	0x15
0	0x18
-1/3	0x1b
-1/2	0x1c
-2/3	0x1d
-1	0x20
-1 (1/3)	0x23
-1 (1/2)	0x24
-1 (2/3)	0x25
-2	0x28
-2 (1/3)	0x2b
-2 (1/2)	0x2c
-2 (2/3)	0x2d
-3	0x30

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_FLASH_COMP

Indicates the flash exposure compensation value.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD020
DataType	prUInt8
DescForms	Enum
Get/Set	Get/Set

Reference:

The retrievable/settable values are the same as those for the exposure compensation value. Refer to Reference of prPTP_DEV_PROP_EXPOSURE_COMP.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_AEB_EXPOSURE_COMP

Indicates the AEB exposure compensation value.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD021
DataType	prUInt8
DescForms	0 (None)
Get/Set	Get

Reference:

The retrievable values are in a range of 0 to +3, which are the same as those for the exposure compensation value. Refer to Reference of prPTP_DEV_PROP_EXPOSURE_COMP.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_AV_OPEN

Indicates the open aperture value.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD023
DataType	prUInt16
DescForms	0 (None)
Get/Set	Get

Reference:

The retrievable values are the same as those for the Av value. Refer to Reference of prPTP_DEV_PROP_AV.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_AV_MAX

Indicates the maximum aperture value.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD024
DataType	prUInt16
DescForms	0 (None)
Get/Set	Get

Reference:

The retrievable values are the same as those for the Av value. Refer to Reference of prPTP_DEV_PROP_AV.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_FOCAL_LENGTH

Indicates the current focal distance set by a capture parameter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD025
DataType	prUInt16
DescForms	0 (None)
Get/Set	Get

Reference:

A value corresponding to the current focal distance multiplied by FocalLengthDenominator (refer to prPTP_DEV_PROP_FOCAL_LENGTH_DENOMI) can be retrieved.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_FOCAL_LENGTH_TELE

Indicates the telescopic focal distance set by a capture parameter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD026
DataType	prUInt16
DescForms	0 (None)
Get/Set	Get

Reference:

A value corresponding to the telescopic focal distance multiplied by FocalLengthDenominator (refer to prPTP_DEV_PROP_FOCAL_LENGTH_DENOMI) can be retrieved.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_FOCAL_LENGTH_WIDE

Indicates the wide-angle focal distance set by a capture parameter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD027
DataType	prUInt16
DescForms	0 (None)
Get/Set	Get

Reference:

A value corresponding to the wide-angle focus distance multiplied by FocalLengthDenominator (refer to prPTP_DEV_PROP_FOCAL_LENGTH_DENOMI) can be retrieved.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_FOCAL_LENGTH_DENOMI

Indicates the focal information multiplier value (FocalLengthDenominator) set by a capture parameter.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD028
DataType	prUInt16
DescForms	0 (None)
Get/Set	Get

Reference:

As for prPTP_DEV_PROP_FOCAL_LENGTH, prPTP_DEV_PROP_FOCAL_LENGTH_TELE, and prPTP_DEV_PROP_FOCAL_LENGTH_WIDE, a value corresponding to each actual value multiplied by prPTP_DEV_PROP_FOCAL_LENGTH_DENOMI can be retrieved.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_CAPTURE_TRANSFER_MODE

Indicates the image transfer mode to be applied at capture.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD029
DataType	prUInt16
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values. Multiple bits can be combined based on the OR value.

Value	Description
0x0000	Not defined.
0x0002	Transfer Entire Image to PC
0x0004	Save Thumbnail Image to Device (A JPEG image will be saved as a normal JPEG file together with the entire image.)
0x0008	Save Entire Image to Device (A JPEG image will be saved as a normal JPEG file together with a thumbnail image.)

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_ZOOM_POS

Indicates the current zoom position when the device uses zoom.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD02A
DataType	prUInt16
DescForms	Range
Get/Set	Get/Set

Reference:

Indicates the current zoom position within the overall zoom range. The specific zoom position is determined by the following data after obtaining the DevicePropDesc data set.

- Current Value: Current zoom position
- Maximum Value (Range-Form): Maximum position in the overall zoom range
- Step Size (Range-Form): Zooming unit (step of zooming)

If the camera device provides electronic zoom, the Maximum Value also includes the number of electronic zoom steps. For information about the DevicePropDesc data set, refer to PR_GetDevicePropDesc() in Chapter 3, "Function References."

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_SUPPORTED_SIZE

Indicates the support size data set consisting of a pair of supported size and image quality. This property is provided for the number of pairs supported by the device.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD02C
DataType	prUInt32 array (refer to Reference)
DescForms	Enum
Get/Set	Get

Reference:

The table below shows DataType (support size data set).

There is an exception for this property. When the DevicePropDesc data set is retrieved using PR_GetDevicePropDesc(), only "Get" will be selected for "Get/Set." However, "Enumeration-Form" will be set in "FORM" and the data will be enumerated in the following data type.

Field	Data type	Description
Num	prUInt32	6 is set.
MainImageHeight	prUInt32	Height of main image. The unit is pixels.
MainImageWidth	prUInt32	Width of main image. The unit is pixels.
MainImageFormat	prUInt32	Type of main image (refer to the table below)
SubImageHeight	prUInt32	Height of thumbnail image. The unit is pixels.
SubImageWidth	prUInt32	Width of thumbnail image. The unit is pixels.
SubImageFormat	prUInt32	Type of thumbnail image (refer to the table below).

The values supported by MainImageFormat and SubImageFormat are shown below.

Value	Description
0x00000000	Not defined.
0x00000001	JPEG SuperFine Large
0x00000002	JPEG SuperFine Middle1
0x00000003	JPEG SuperFine Middle2
0x00000004	JPEG SuperFine Middle
0x00000005	JPEG SuperFine Small
0x00000006	JPEG Fine Large
0x00000007	JPEG Fine Middle1
0x00000008	JPEG Fine Middle2
0x00000009	JPEG Fine Middle
0x0000000A	JPEG Fine Small
0x0000000B	JPEG Normal Large
0x0000000C	JPEG Normal Middle1
0x0000000D	JPEG Normal Middle2
0x0000000E	JPEG Normal Middle
0x0000000F	JPEG Normal Small
0x00000010	CRW Large
0x00000011	JPEG SuperFine Middle3
0x00000012	JPEG Fine Middle3
0x00000013	JPEG Normal Middle3
0x00000014	CR2 Large

➢ prPTP_DEV_PROP_SUPPORTED_THUMB_SIZE

Indicates information regarding the thumbnail size (position) supported by the device.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD02D
DataType	prUInt32 array (refer to "Reference")
DescForms	0 (None)
Get/Set	Get

Reference:

The table below shows DataType (thumbnail size data set).

Field	Data type	Description
Num	prUInt32	Length of thumbnail size data set.
Bottom	prUInt32	Bottom position of thumbnail.
Left	prUInt32	Left position of thumbnail.
Right	prUInt32	Right position of thumbnail.
Top	prUInt32	Top position of thumbnail.

➢ prPTP_DEV_PROP_FIRMWARE_VERSION

Indicates the version of the camera device firmware.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD031
DataType	prUInt32
DescForms	0 (None)
Get/Set	Get

Reference:

Version information consists of a four-digit number, where each of the four bytes represents a single value. For example, the hexadecimal number "01 00 00 00" indicates version "1.0.0.0."

➢ prPTP_DEV_PROP_CAMERA_MODEL_NAME

Indicates the camera model.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD032
DataType	String (refer to "Reference")
DescForms	0 (None)
Get/Set	Get

Reference:

DataType (String type) is shown below.

Data field	Data type	Description
NumChars	prUInt8	Number of characters in StringChars
StringChars	prWChar array	Unicode character string (consisting of up to 255 characters, ending with a NULL).

➢ prPTP_DEV_PROP_OWNER_NAME

Indicates the owner name.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD033
DataType	prUInt8 array (refer to "Reference")
DescForms	0 (None)
Get/Set	Get/Set

Reference:

DataType (prUInt8 array) is shown below.

Data field	Data type	Description
NumElements	prUInt32	Number of characters in OwnerNameChars
OwnerNameChars	prUInt8 array	Character string indicating the owner name, ending with a NULL.

➢ prPTP_DEV_PROP_CAMERA_TIME

Indicates the current time information in the device.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD034
DataType	prUInt32
DescForms	0 (None)
Get/Set	Get/Set

Reference:

The time set here corresponds to the number of elapsed seconds since 00 hours 00 minutes 00 seconds on January 1, 1970 based on UTC (Universal Time Coordinated).

➢ prPTP_DEV_PROP_CAMERA_OUTPUT

Indicates the destination of image signal output in the Viewfinder mode.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD036
DataType	prUInt8
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Value	Description
0x0000	Not defined.
0x0001	LCD
0x0002	Video OUT
0x0003	Off

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_DISP_AV

Indicates how to display the Av value as described in "Reference".

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD037
DataType	prUInt16
DescForms	0 (None)
Get/Set	Get

Reference:

Indicates the retrievable values. Associated with prPTP_DEV_PROP_AV_OPEN_APEX.

Value	Description
0x0000	1/3 Level
0x0001	A value corresponding to one-tenth the value retrieved by prPTP_DEV_PROP_AV_OPEN_APEX is displayed.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_AV_OPEN_APEX

Indicates the open aperture value.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD038
DataType	prUInt16
DescForms	0 (None)
Get/Set	Get

Reference:

Indicates the retrievable values. The value varies depending upon the display method set by prPTP_DEV_PROP_DISP_AV.

Value	Description
(Value retrieved when prPTP_DEV_PROP_DISP_AV == 0x00)	The camera's APEX value is retrieved.
(Value retrieved when prPTP_DEV_PROP_DISP_AV == 0x01)	A value corresponding to 10 times the displayed Av value is retrieved.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_EZOOM_SIZE

Indicates the horizontal size of image to be cut out from CCD image using electronic zoom.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD039
DataType	prInt16
DescForms	0 (None)
Get/Set	Get

Reference:

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_ML_SPOT_POS

Indicates the spot metering position.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD03A
DataType	prUInt16
DescForms	0 (None)
Get/Set	Get

Reference:

Indicates the retrievable values.

Value	Description
0	MlSpotPosCenter
1	MlSpotPosAfLink

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_DISP_AV_MAX

Indicates how to display the maximum Av value as described in "Reference".

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD03B
DataType	prUInt16
DescForms	0 (None)
Get/Set	Get

Reference:

Indicates the retrievable values. Associated with prPTP_DEV_PROP_AV_MAX_APEX.

Value	Description
0x0000	1/3 Level
0x0001	A value corresponding to one-tenth the value retrieved by prPTP_DEV_PROP_AV_MAX_APEX is displayed.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_AV_MAX_APEX

Indicates the minimum aperture value.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD03C
DataType	prUInt16
DescForms	0 (None)
Get/Set	Get

Reference:

Indicates the retrievable values. The value varies depending upon the display method set by prPTP_DEV_PROP_AV_MAX_APEX.

Value	Description
(Value retrieved when prPTP_DEV_PROP_AV_MAX_APEX == 0x00)	The camera's APEX value is retrieved.
(Value retrieved when prPTP_DEV_PROP_AV_MAX_APEX == 0x01)	A value corresponding to 10 times the displayed Av value is retrieved.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_EZOOM_START_POS

Indicates the zoom position at which the electronic zoom range starts.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD03D
DataType	prUInt16
DescForms	0 (None)
Get/Set	Get

Reference:

The retrieved value varies depending upon the camera model. If this value is 0xFF, it means the camera has no electronic zoom.

> Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_FOCAL_LENGTH_OF_TELE

Indicates the focal distance at the optical telescopic end.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD03E
DataType	prUInt16
DescForms	0 (None)
Get/Set	Get

Reference:

The retrieved value varies depending upon the camera model.

> Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_EZOOM_SIZE_OF_TELE

Indicates the horizontal size of image to be cut out from CCD image at the telescopic end of the electronic zoom range.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD03F
DataType	prUInt16
DescForms	0 (None)
Get/Set	Get

Reference:

The retrieved value varies depending upon the camera model.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_PHOTO_EFFECT

Indicates the photo effect.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD040
DataType	prUInt16
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Value	Description
0x0000	Off
0x0001	Vivid
0x0002	Neutral
0x0003	Soft
0x0004	Sepia
0x0005	Monochrome

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_AF_LIGHT

Indicates the on/off of AF-assist light.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD041
DataType	prUInt16
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Value	Description
0x0000	Off
0x0001	On

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_FLASH_QUANTITY

Indicates the number of flash levels that can be set in the manual mode.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD042
DataType	prUInt16
DescForms	Enum
Get/Set	Get/Set

Reference:

Among the retrievable/settable values, 0 indicates full flashing, while FlashLevelCount - 1 indicates the minimum flashing.

Note: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_ROTATION_ANGLE

Indicates the angle of rotation detected by the gravity sensor.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD043
DataType	prUInt16
DescForms	0 (None)
Get/Set	Get

Reference:

Indicates the retrievable values.

Value	Description
0x0000	0 Degree
0x0001	90 Degrees
0x0002	180 Degrees
0x0003	270 Degrees
0xffff	None

Note 1: "None" is set on models that do not support this function.

Note 2: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_ROTATION_SENCE

Indicates whether the gravity sensor is enabled or disabled.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD044
DataType	prUInt16
DescForms	Enum
Get/Set	Get/Set

Reference:

Indicates the retrievable/settable values.

Value	Description
0x0000	Enable
0x0001	Disable
0xffff	None

Note 1: "None" is set on models that do not support this function.

Note 2: This property is valid only in the remote release control mode. If the camera is not in the remote release control mode, the value of this property is invalid.

➢ prPTP_DEV_PROP_IMEGE_FILE_SIZE

Indicates the image file size supported by the camera.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD048
DataType	prUInt32 array (refer to "Reference")
DescForms	0 (None)
Get/Set	Get

Reference:

The table below shows DataType.

0 KB is set as the file size for images of types not supported by the camera.

Field	Data type	Description
Num	prUInt32	19 is set.
JPEG SuperFine Large	prUInt32	File Size (KB)
JPEG SuperFine Middle1	prUInt32	File Size (KB)
JPEG SuperFine Middle2	prUInt32	File Size (KB)
JPEG SuperFine Middle	prUInt32	File Size (KB)
JPEG SuperFine Small	prUInt32	File Size (KB)
JPEG Fine Large	prUInt32	File Size (KB)
JPEG Fine Middle1	prUInt32	File Size (KB)
JPEG Fine Middle2	prUInt32	File Size (KB)
JPEG Fine Middle	prUInt32	File Size (KB)
JPEG Fine Small	prUInt32	File Size (KB)
JPEG Normal Large	prUInt32	File Size (KB)
JPEG Normal Middle1	prUInt32	File Size (KB)
JPEG Normal Middle2	prUInt32	File Size (KB)
JPEG Normal Middle	prUInt32	File Size (KB)
JPEG Normal Small	prUInt32	File Size (KB)
CRW Large	prUInt32	File Size (KB)
JPEG SuperFine Middle3	prUInt32	File Size (KB)
JPEG Fine Middle3	prUInt32	File Size (KB)
JPEG Normal Middle3	prUInt32	File Size (KB)

➢ prPTP_DEV_PROP_CAMERA_MODEL_ID

Indicates the camera model ID.

DevicePropDesc information

DevicePropDesc field	Setting
DevicePropCode	0xD049
DataType	prUInt32
DescForms	0 (None)
Get/Set	Get

Reference:

The same as the model ID in the DeviceInfo data set that can be retrieved using PR_GetDeviceList().

6 Events

This chapter describes the events supported by the PS-ReC SDK.

For the procedure to retrieve event data and the structure of event data, refer to PR_SetEventCallBack()/PR_ClearEventCallBack() in Chapter 3, "Function References" or prSetEventCB() in Chapter 4, "Structure and Callback Function Definitions."

Note: The types of events that are supported may differ depending upon the camera model. Before retrieving events, retrieve the DeviceInfo data set using PR_GetDeviceInfo(). The event codes that can be used for the applicable camera device will be enumerated in the EventsSupported field.

➢ prPTP_DEVICE_PROP_CHANGED

Notifies that a device property has been changed. For information about DevicePropCode, refer to Chapter 5, "Device Properties."

Event information

EventData field	Setting
EventCode	0x4006
Parameter1	DevicePropCode
Parameter2	None
Parameter3	None

➢ prPTP_CAPTURE_COMPLETED

Notifies that the capture has finished.

Event information

EventData field	Setting
EventCode	0x400D
Parameter1	Error code
Parameter2	None
Parameter3	None

➢ prPTP_SHUTDOWN_CF_GATE_WAS_OPENED

Notifies that the device has shut down due to the opening of the CF/SD card cover.

Event information

EventData field	Setting
EventCode	0xC001
Parameter1	None
Parameter2	None
Parameter3	None

➢ prPTP_RESET_HW_ERROR

Notifies that the device has generated a hardware error.

Event information

EventData field	Setting
EventCode	0xC005
Parameter1	Error code
Parameter2	None
Parameter3	None

Reference:

The error codes defined in "prError.h" are set.

➢ prPTP_ABORT_PC_EVF

Notifies that the Viewfinder mode has been cancelled.

Event information

EventData field	Setting
EventCode	0xC006
Parameter1	None
Parameter2	None
Parameter3	None

➢ prPTP_ENABLE_PC_EVF

Notifies that the Viewfinder mode has been enabled.

Event information

EventData field	Setting
EventCode	0xC007
Parameter1	None
Parameter2	None
Parameter3	None

➢ prPTP_FULL_VIEW_RELEASED

This event is notified from the camera to the PC after remote capture to communicate the transfer timing of main image file data.

Event information

EventData field	Setting
EventCode	0xC008
Parameter1	Object handle
Parameter2	None
Parameter3	None

Reference:

Note: By specifying this event code and the object handle of Parameter 1 in the parameters of PR_GetReleasedData(), the captured main image can be transferred to the PC. For details, refer to PR_GetReleasedData() in Chapter 3, "Function References."

➢ prPTP_THUMBNAIL_RELEASED

This event is notified from the camera to the PC after remote capture to communicate the transfer timing of thumbnail image file data.

Event information

EventData field	Setting
EventCode	0xC009
Parameter1	Object handle
Parameter2	None
Parameter3	None

Reference:

Note: By specifying this event code and the object handle of Parameter 1 in the parameters of PR_GetReleasedData(), a thumbnail of the captured image can be transferred to the PC. For details, refer to PR_GetReleasedData() in Chapter 3, "Function References."

➢ prPTP_CHANGE_BATTERY_STATUS

Notifies that the power condition of the camera has changed.

Event information

EventData field	Setting
EventCode	0xC00A
Parameter1	Battery type (refer to "Reference")
Parameter2	Battery level (refer to "Reference")
Parameter3	None

Reference:

Indicates the battery type in Parameter 1.

Value	Description
0x01	AC power supply
0x02	Lithium battery
0x03	Nickel hydride battery
0x04	Nickel cadmium battery
0x05	Alkaline manganese battery

Indicates the battery level in Parameter 2.

Value	Description
0x01	NORMAL
0x02	WARNING_LV1
0x03	EMERGENCY
0x04	WARNING_LV0

➢ prPTP_PUSHED_RELEASE_SW

Notifies that the user has pressed the release switch on the camera when the camera was in the remote release control mode and remote capture was enabled.

Event information

EventData field	Setting
EventCode	0xC00B
Parameter1	None
Parameter2	None
Parameter3	None

Reference:

Note: After the user presses the switch, the camera does not execute the capture. It only issues this event. After receiving this event, the client should call PR_RC_Release() to perform remote capture.

➢ prPTP_RC_PROP_CHANGED

Notifies that a group of properties relating to release control have been changed as a result of executing PR_InitiateReleaseControl() and the like.

Event information

EventData field	Setting
EventCode	0xC00C
Parameter1	None
Parameter2	None
Parameter3	None

Reference:

After receiving this event data, the client should retrieve all of the properties relating to release control again.

➢ prPTP_RC_ROTATION_ANGLE_CHANGED

Notifies that the angle of rotation of the camera has been changed in the remote release control mode.

Event information

EventData field	Setting
EventCode	0xC00D
Parameter1	Angle of rotation (refer to "Reference")
Parameter2	None
Parameter3	None

Reference:

Indicates the angle of rotation.

Value	Description
0x00	0 Degree
0x01	90 Degrees
0x02	180 Degrees
0x03	240 Degrees
0xffffffff	Invalid

➢ prPTP_RC_CHANGED_BY_CAM_UI

Notifies that an operation control on the camera has been operated in the remote release control mode.

Event information

EventData field	Setting
EventCode	0xC00E
Parameter1	None
Parameter2	None
Parameter3	None

Reference:

Note: After receiving this event, the client should issue PR_RC_GetChangedReleaseParamsList() and retrieve the device property information that was changed.

➢ prCAL_SHUTDOWN

Notifies that a shutdown event has occurred as a result of disconnection between the PC and camera device or the turning off of the camera power.

Event information

EventData field	Setting
EventCode	0xD001
Parameter1	None
Parameter2	None
Parameter3	None

Note: This event is not a camera device event, so it is not enumerated in the EventsSupported field of the DeviceInfo data set.

7 Sample

Simple sample code using the PS-ReC SDK is shown below.

Other samples are also available in the medium containing the PS-ReC SDK. Refer to the Readme.txt file of each sample for details.

Note:	When executing a sample, copy the necessary modules to the folder in accordance with "How to Use PS-ReC SDK Modules" in Chapter 1, "Getting Started."

Sample

This sample covers the steps to initialize the PS-ReC SDK, start Viewfinder, and perform remote capture. It also indicates how to use the WIA/STI interfaces based on the camera device enumeration obtained by PR_GetDeviceList().

For details on the headers/libraries of the WIA or STI interface or the interface itself, refer to the Help and samples provided in:

- "Microsoft Platform SDK" (Microsoft)
- "Windows Driver Development Kit" (Microsoft)

Note: When accessing a camera using the STI interface, you must disconnect the camera and then access the camera again by executing PR_DisconnectCamera() provided by the PS-ReC SDK, as shown in the sample.

```
/* Sample */
#include <windows.h>
#include <stdio.h>

/* STI/WIA header */
/* Use Platform SDK */
#include  <objbase.h>
#include  <sti.h>
#include  <wia.h>

/* PS-ReC SDK header */
#include "prAPI.h"

#define         MY_BUFFER_SIZE       20480L     /* 20KB */

typedef struct {
     prUInt32       ContainerLength;
     prUInt16       ContainerType;
     prUInt16       Code;
     prUInt32       TransactionID;
     prUInt32       Parameter[prANY];
} EVENT_GENERIC_CONTAINER;
```

```
prObjectHandle  g_ObjectHandle = 0L;

prResponse prSTDCALL  MyEventCallbackFunction(
     prHandle   CameraHandle,
     prContext  Context,
     prVoid          *pEventData
);

prResponse prSTDCALL  MyGetFileDataFunction(
     prHandle       CameraHandle,
     prObjectHandle  ObjectHandle,
     prContext      Context,
     prProgress     *pProgress
);

prResponse prSTDCALL  MyViewFinderFunction(
     prHandle   CameraHandle,
     prContext  Context,
     prUInt32   Size,
     prVoid          *pVFData
);

prVoid    IsSupportedRCandEVF(
     prUInt8         *pDeviceInfo,
     prBoolean  *pIsRC,
     prBoolean  *pIsEVF
);

prBoolean  IsSupportedCapTransMode(
     prUInt8         *pDeviceInfo
);

prBoolean  IsSupportedMode(
     prUInt8         *pDevicePropDesc
);

prBoolean IsSupportedOS(
     prBoolean  *pIsXP
);

HRESULT WIA_Function(
     prDeviceInfoTable    *pDeviceInfoTable
);
```

```
HRESULT STI_Function(
     prDeviceInfoTable    *pDeviceInfoTable
);

/* StiCreateInstance() type definition */
typedef BOOL (CALLBACK*  LP_STI_CREATE_INSTANCE)(
     HINSTANCE,
     DWORD,
     PSTI*,
     LPUNKNOWN
);

void main()
{
     prResponse      err = prOK;
     prHandle        hCamera = 0L;
     prUInt32        BufferSize = 0L, DataSize = 0L;
     prUInt8         Buffer[MY_BUFFER_SIZE];
     prUInt8         DeviceListBuffer[MY_BUFFER_SIZE];
     prDeviceList    *pDeviceList = NULL;
     prContext       Context = 0;
     prBoolean       IsRC = FALSE, IsEVF = FALSE;
     prBoolean       IsCapTransMode = FALSE, IsMode = FALSE;
     prUInt16        CapTransMode = 0;
     prBoolean       IsOS = FALSE, IsXP = FALSE;
     HRESULT         hr = S_OK;

     /* Determine the OS */
     IsOS = IsSupportedOS(&IsXP);
     if (IsOS == FALSE) {
          /* The OS environment is not supported by the PS-ReC SDK */
          return;
     }

     /* Start the PS-ReC SDK */
     err = PR_StartSDK();
     if (err != prOK) {
          return;
     }

     /* Enumerate camera devices  */
     BufferSize = MY_BUFFER_SIZE;
     memset(Buffer, 0, MY_BUFFER_SIZE);
     err = PR_GetDeviceList(&BufferSize,  (prDeviceList *)Buffer);
```

```
if (err != prOK) {
     goto Finish;
}
memset(DeviceListBuffer, 0, MY_BUFFER_SIZE);
memcpy(DeviceListBuffer, Buffer, BufferSize);
pDeviceList = (prDeviceList    *)DeviceListBuffer;
if (pDeviceList->NumList == 0L) {
     /* No camera is connected */
     goto Finish;
}

/* Create a camera handle */
err = PR_CreateCameraObject(&(pDeviceList->DeviceInfo[0]), &hCamera);
if (err != prOK) {
     goto Finish;
}

/* Register an event callback function  */
err = PR_SetEventCallBack(
     hCamera,
     0,
     &MyEventCallbackFunction
);
if (err != prOK) {
     goto Finish;
}

/* Connect a camera device */
err = PR_ConnectCamera(hCamera);
if (err != prOK) {
     goto Finish;
}

/* Retrieve the DeviceInfo data set */
BufferSize = MY_BUFFER_SIZE;
memset(Buffer, 0, MY_BUFFER_SIZE);
err = PR_GetDeviceInfo(hCamera,&BufferSize, Buffer);
if (err != prOK) {
     goto Finish;
}

/* Check if remote capture and Viewfinder are supported */
IsSupportedRCandEVF(Buffer, &IsRC, &IsEVF);
if (IsRC == FALSE) {
```

```
        goto Finish;
    }

    /* Start the remote release control mode */
    err = PR_InitiateReleaseControl(hCamera);
    if (err != prOK) {
        goto Finish;
    }

    /* Start Viewfinder, if supported */
    if (IsEVF == TRUE) {
        err = PR_RC_StartViewFinder(
                    hCamera,
                    0,
                    &MyViewFinderFunction
                );
        if (err != prOK) {
            goto Finish;
        }
    }

    /* Supported device property was changed,*/
    /* so retrieve the DeviceInfo data set again  */
    BufferSize = MY_BUFFER_SIZE;
    memset(Buffer, 0, MY_BUFFER_SIZE);
    err = PR_GetDeviceInfo(
        hCamera,
        &BufferSize,
        Buffer
    );
    if (err != prOK) {
        goto Finish;
    }

    IsCapTransMode = IsSupportedCapTransMode(Buffer);
    /* Supported */
    if (IsCapTransMode == TRUE) {
        /* Retrieve the DevicePropDesc data set */
        BufferSize = MY_BUFFER_SIZE;
        memset(Buffer, 0, MY_BUFFER_SIZE);
        err = PR_GetDevicePropDesc(
            hCamera,
            prPTP_DEV_PROP_CAPTURE_TRANSFER_MODE,
            &BufferSize,
```

```
            Buffer
        );
        if (err != prOK) {
            goto Finish;
        }
        /* Check if JPEG is supported as a file type at capture
            */

        /* (Also check if device properties can be set )    */
        IsMode = IsSupportedMode(Buffer);
        if (IsMode == TRUE) {
            /* Transfer main image only and save it to the memory card*/
            DataSize = (prUInt32)sizeof(prUInt16);
            CapTransMode = 0x0003;
            err = PR_SetDevicePropValue(
                hCamera,
                prPTP_DEV_PROP_CAPTURE_TRANSFER_MODE,
                DataSize,
                &CapTransMode
            );
            if (err != prOK) {
                goto Finish;
            }
        }
    }

    /* Remote capture */
    err = PR_RC_Release(hCamera);
    if (err != prOK) {
        goto Finish;
    }

    /* Capture main image */
    err = PR_RC_GetReleasedData(
        hCamera,
        g_ObjectHandle,
        prPTP_FULL_VIEW_RELEASED,
        1024L,
        0,
        &MyGetFileDataFunction
    );
    if (err != prOK) {
        goto Finish;
    }
```

```
     /* Capture thumbnail */
     err = PR_RC_GetReleasedData(
          hCamera,
          g_ObjectHandle,
          prPTP_THUMBNAIL_RELEASED,
          1024L,
          0,
          &MyGetFileDataFunction
     );

     /* Termination process */
Finish:
     if (hCamera != 0L) {
          if (IsEVF == TRUE) {
               /* Exit Viewfinder */
               (prVoid)PR_RC_TermViewFinder(hCamera);
          }
          if (IsRC == TRUE) {
               /* End the remote release control mode  */
               (prVoid)PR_TerminateReleaseControl(hCamera);
          }
          /* If the memory card contains any image captured with XP, */
          if (IsXP == TRUE) {
               /* access the camera using WIA */
               /* Even if the camera device is connected via the PS-ReC SDK,*/
               /* you can still use WIA. */
               /* However, be sure to end the remote release control mode.
*/
               hr = WIA_Function(&(pDeviceList->DeviceInfo[0]));
          }
          /* Disconnect the camera device */
          (prVoid)PR_DisconnectCamera(hCamera);
          /* Release the event callback function  */
          (prVoid)PR_ClearEventCallBack(hCamera);
          /* Delete the camera handle*/
          (prVoid)PR_DestroyCameraObject(hCamera);
     }
     /* End the PS-ReC SDK  */
     (prVoid)PR_FinishSDK();

     /* If 98SE, 2000 SP4 or Me is used, */
     if (IsXP == FALSE) {
          /* disconnect the camera device using the PS-ReC SDK first, */
```

```
          /* and then access the camera via STI.  */
          hr = STI_Function(&(pDeviceList->DeviceInfo[0]));
     }

     return;
}

prUInt32 prSTDCALL MyEventCallbackFunction (
     prHandle  CameraHandle,
     prContext  Context,
     prVoid*   pEventData
)
{
     EVENT_GENERIC_CONTAINER        *pEventDataTemp;

     pEventDataTemp = (EVENT_GENERIC_CONTAINER *)pEventData;

     /* Display event codes*/
     printf("Event Code = 0x%x\n", pEventDataTemp->Code);

     if (pEventDataTemp->Code == prPTP_FULL_VIEW_RELEASED) {
          g_ObjectHandle = (prObjectHandle)pEventDataTemp->Parameter[0];
     }

     return prOK;
}

prResponse prSTDCALL  MyGetFileDataFunction(
     prHandle     CameraHandle,
     prObjectHandle ObjectHandle,
     prContext    Context,
     prProgress*   pProgress
)
{
     /* Display progress */
     if (pProgress->lMessage == prMSG_DATA) {
          printf("Progress = %d\n", pProgress->lPercentComplete);
     }

     return prOK;
}

prResponse prSTDCALL  MyViewFinderFunction(
     prHandle  CameraHandle,
```

```
      prContext  Context,
      prUInt32   Size,
      prVoid          *pVFData
)
{
      /* Display the Viewfinder data size */
      printf("View Finder Data Size = %d\n", Size);

      return prOK;
}

prVoid    IsSupportedRCandEVF(
      prUInt8    *pDeviceInfo,   /* (IN)  DeviceInfo data set */
      prBoolean  *pIsRC,
      prBoolean  *pIsEVF
)
{
      prUInt8         *pDeviceInfoTmp;
      prUInt8         bNum;
      prUInt32   dwNum, i;
      prOperationCode wOperation;
      /* Flag indicating whether or not an operation is supported */
      prBoolean   fInitiateRC, fTerminateRC;
      prBoolean   fCapture, fInitiateEVF, fTerminateEVF;

      fInitiateRC = FALSE;
      fTerminateRC = FALSE;
      fCapture = FALSE;
      fInitiateEVF  = FALSE;
      fTerminateEVF = FALSE;

      pDeviceInfoTmp = pDeviceInfo;

/* Move the pointer to "Supported operations" */
      /* Standard version*/
      /* Vendor extendedID */
      /* Vendor extended version */
      pDeviceInfoTmp +=  sizeof(prUInt16)
                          + sizeof(prUInt32)
                          + sizeof(prUInt16);
      /* Vendor extended information */
      bNum = *((prUInt8 *)pDeviceInfoTmp);
      pDeviceInfoTmp +=  sizeof(prUInt8)
                            + sizeof(prWChar) * bNum;
```

```
     /* Function modes*/
     pDeviceInfoTmp +=  sizeof(prUInt16);

     /* Supported operations */
     dwNum = *((prUInt32 *)pDeviceInfoTmp);    /* Number of elements */
     pDeviceInfoTmp +=  sizeof(prUInt32);
     /* Loop for the number of elements  */
     for (i = 0L; i < dwNum; i++) {
          wOperation = *((prOperationCode *)pDeviceInfoTmp);
          if (wOperation == prPTP_INITIATE_RELEASE_CONTROL) {
               fInitiateRC = TRUE;
          }
          if (wOperation == prPTP_TERMINATE_RELEASE_CONTROL) {
               fTerminateRC = TRUE;
          }
          if (wOperation == prPTP_RC_CAPTURE) {
               fCapture = TRUE;
          }
          if (wOperation == prPTP_RC_INITIATE_VIEW_FINDER) {
               fInitiateEVF = TRUE;
          }
          if (wOperation == prPTP_RC_TERMINATE_VIEW_FINDER) {
               fTerminateEVF = TRUE;
          }
          pDeviceInfoTmp += sizeof(prOperationCode);
     }

/* The following information is not checked */
     /* Supported events */
     /* Supported device properties */
     /* Supported captured image types */
     /* Supported image types */
     /* Company information */
     /* Model name */
     /* Device version*/
     /* Serial number*/

     if (  (fInitiateRC == TRUE)
          && (fTerminateRC == TRUE)
          && (fCapture == TRUE)) {
          /* Remote capture is supported */
          *pIsRC = TRUE;
     }
     else {
```

```
          /* Remote capture is not supported */
          *pIsRC = FALSE;
     }

     if (  (fInitiateEVF == TRUE)
          && (fTerminateEVF == TRUE)) {
          /* Viewfinder is supported */
          *pIsEVF = TRUE;
     }
     else {
          /* Viewfinder is not supported */
          *pIsEVF = FALSE;
     }
}

prBoolean  IsSupportedCapTransMode(
     /* (IN)  DeviceInfo data set */
     prUInt8*   pDeviceInfo
)
{
     prUInt8         *pDeviceInfoTmp;
     prUInt8         bNum;
     prUInt32   dwNum, i;
     prptpDevicePropCode  wDeviceProp;

     pDeviceInfoTmp = pDeviceInfo;

/* Move the pointer to "Supported device properties" */
     /* Standard version*/
     /* Vendor extended ID */
     /* Vendor extended version */
     pDeviceInfoTmp +=  sizeof(prUInt16)
                      + sizeof(prUInt32)
                      + sizeof(prUInt16);
     /* Vendor extended information */
     bNum = *((prUInt8 *)pDeviceInfoTmp);
     pDeviceInfoTmp +=  sizeof(prUInt8)
                      + sizeof(prWChar) * bNum;
     /* Function modes*/
     pDeviceInfoTmp +=  sizeof(prUInt16);
     /* Supported operations */
     dwNum = *((prUInt32 *)pDeviceInfoTmp);
     pDeviceInfoTmp +=  sizeof(prUInt32)
                      + sizeof(prUInt16) * dwNum;
```

```
     /* Supported events */
     dwNum = *((prUInt32 *)pDeviceInfoTmp);
     pDeviceInfoTmp +=  sizeof(prUInt32)
                     + sizeof(prUInt16) * dwNum;

     /* Supported device properties */
     dwNum = *((prUInt32 *)pDeviceInfoTmp);    /* Number of elements */
     pDeviceInfoTmp += sizeof(prUInt32);
     for (i = 0L; i < dwNum; i++) {
          wDeviceProp = *((prptpDevicePropCode *)pDeviceInfoTmp);
          if (wDeviceProp == prPTP_DEV_PROP_CAPTURE_TRANSFER_MODE) {
               /* Image type at capture is supported */
               return TRUE;
          }
          pDeviceInfoTmp += sizeof(prptpDevicePropCode);
     }

/* The following information is not checked */
     /* Supported captured image types */
     /* Supported image types */
     /* Company information */
     /* Model name */
     /* Device version*/
     /* Serial number*/

     /* Image type at capture is not supported */
     return FALSE;
}

prBoolean  IsSupportedMode(
     /* (IN)  DevicePropDesc data set */
     prUInt8         *pDevicePropDesc
)
{
     prUInt8    *pDevicePropDescTemp;
     prptpDevicePropCode DevicePropCode;
     prUInt8         bGetSet, bFormFlag;
     prUInt16   wDataType;
     prUInt16   wNum, i;
     prUInt16   wMode;

     pDevicePropDescTemp = pDevicePropDesc;

     /* Device Property Code */
```

```
DevicePropCode = *((prptpDevicePropCode *)pDevicePropDescTemp);
if (DevicePropCode != prPTP_DEV_PROP_CAPTURE_TRANSFER_MODE) {
     /* Error, if not image type */
     return FALSE;
}
pDevicePropDescTemp += sizeof(prptpDevicePropCode);

/* DataType */
wDataType = *((prUInt16 *)pDevicePropDescTemp);
if (wDataType != 0x0004) {
     /* Error, if not prUInt16 */
     return FALSE;
}
pDevicePropDescTemp += sizeof(prUInt16);

/* GetSet */
bGetSet = *((prUInt8 *)pDevicePropDescTemp);
if (bGetSet != 0x01) {
     /* Error, if not GetSet */
     return FALSE;
}
pDevicePropDescTemp += sizeof(prUInt8);

/* FactoryDefaultValue */
/* CurrentValue */
pDevicePropDescTemp +=  sizeof(prUInt16)
                      + sizeof(prUInt16);

/* FormFlag */
bFormFlag = *((prUInt8 *)pDevicePropDescTemp);
if (bFormFlag != 0x02) {
     /* Error, if not Enumeration-Form */
     return FALSE;
}
pDevicePropDescTemp += sizeof(prUInt8);

/* Enumeration-Form */
wNum = *((prUInt16 *)pDevicePropDescTemp);
pDevicePropDescTemp += sizeof(prUInt16);
for (i = 0; i < wNum; i++) {
     wMode = *((prUInt16 *)pDevicePropDescTemp);
     if (wMode == 0x0003) {
          /* Main image transfer and transfer settings are possible */
          return TRUE;
```

```
            }
            pDevicePropDescTemp += sizeof(prUInt16);
      }

      return FALSE;
}

/* TRUE: OS supported by the PS-ReC SDK, FALSE: OS not supported by the PS-ReC
SDK */
prBoolean IsSupportedOS(
      prBoolean* pIsXP/* TRUE:XP, FALSE:98SE,2000 SP4,ME */
)
{
      /* Retrieve applicable OS*/
      OSVERSIONINFOEX OsVerInfoEx;
      memset(&OsVerInfoEx, 0, sizeof(OSVERSIONINFOEX));
      OsVerInfoEx.dwOSVersionInfoSize = sizeof(OSVERSIONINFOEX);
      /* Retrieve version information */
      if (!GetVersionEx((OSVERSIONINFO *)&OsVerInfoEx)) {
            return FALSE;
      }
      // Generate class by version
      if (OsVerInfoEx.dwPlatformId == VER_PLATFORM_WIN32_NT) {
            /* XP */
            if (  (OsVerInfoEx.dwMajorVersion == 5)
                  && (OsVerInfoEx.dwMinorVersion == 1)
              ) {
                  *pIsXP = TRUE;
                  return TRUE;
            }
            else if (  (OsVerInfoEx.dwMajorVersion == 5)
                        && (OsVerInfoEx.dwMinorVersion == 0)
                       ) {
                  /* 2000 SP4 */
                  if (OsVerInfoEx.wServicePackMajor >= 4) {
                        *pIsXP = FALSE;
                        return TRUE;
                  }
            }
      }
      else if (OsVerInfoEx.dwPlatformId == VER_PLATFORM_WIN32_WINDOWS) {
            if (  (OsVerInfoEx.dwMajorVersion == 4)
                  && (OsVerInfoEx.dwMinorVersion == 10)
              ) {
```

```
                /* 98SE */
                if (OsVerInfoEx.szCSDVersion[1] == 'A') {
                     *pIsXP = FALSE;
                     return TRUE;
                }
          }
          /* ME */
          else if (   (OsVerInfoEx.dwMajorVersion == 4)
                      && (OsVerInfoEx.dwMinorVersion == 90)
            ) {
                *pIsXP = FALSE;
                return TRUE;
          }
     }
     /* Other*/
     return FALSE;
}

HRESULT WIA_Function(
     prDeviceInfoTable*   pDeviceInfoTable
)
{
     HRESULT         hr = S_OK;
     IWiaDevMgr*     pIWiaDevMgr = NULL;
     IWiaItem*  pIWiaRootItem = NULL;
     BSTR       bstrDevName = NULL;

     hr = CoInitializeEx(NULL, COINIT_APARTMENTTHREADED);
     if (FAILED(hr)) {
          return hr;
     }

     bstrDevName = SysAllocString(pDeviceInfoTable->DeviceInternalName);
     if (!bstrDevName) {
          hr = E_FAIL;
          goto Finish;
     }

     hr = CoCreateInstance(
                CLSID_WiaDevMgr,
                NULL,
                CLSCTX_LOCAL_SERVER,
                IID_IWiaDevMgr,
                (void**)&pIWiaDevMgr
```

```
      );
     if (FAILED(hr)) {
          goto Finish;
     }

     /* Connect to a camera device */
     hr = pIWiaDevMgr->CreateDevice(bstrDevName, &pIWiaRootItem);
     if (FAILED(hr)) {
          goto Finish;
     }

     /* Perform WIA processing */

Finish:
     if (pIWiaRootItem) {
          (void)pIWiaRootItem->Release();
     }

     if (pIWiaDevMgr) {
          pIWiaDevMgr->Release();
     }

     if (bstrDevName) {
          SysFreeString(bstrDevName);
     }

     CoUninitialize();

     return hr;
}

HRESULT STI_Function(
     prDeviceInfoTable*   pDeviceInfoTable
)
{
     HRESULT                         hr = S_OK;
     HINSTANCE                  hSti = NULL;
     LP_STI_CREATE_INSTANCE     pStiCreateInstance = NULL;
     PSTI                       pIStillImage = NULL;
     PSTIDEVICE                 pIStiDevice = NULL;

     hSti = LoadLibrary("STI.DLL");
     if (!hSti) {
          return E_FAIL;
```

```
    }

    pStiCreateInstance = (LP_STI_CREATE_INSTANCE)GetProcAddress(
        hSti,
        "StiCreateInstance"
    );
    if (!pStiCreateInstance) {
        hr = E_FAIL;
        goto Finish;
    }

    hr = CoInitializeEx(NULL, COINIT_APARTMENTTHREADED);
    if (FAILED(hr)) {
        goto Finish;
    }

    hr = pStiCreateInstance(
        GetModuleHandle(NULL),
        STI_VERSION,
        &pIStillImage,
        NULL
    );
    if (FAILED(hr)) {
        goto Finish;
    }

    /* Connect to a camera device */
    hr = pIStillImage->CreateDevice(
        pDeviceInfoTable->DeviceInternalName,
        STI_DEVICE_CREATE_BOTH,
        &pIStiDevice,
        NULL
    );

    /* Perform STI processing */

Finish:
    if (pIStillImage) {
        (void)pIStillImage->Release();
        pIStillImage = NULL;
    }

    CoUninitialize();
```

```
    if (hSti) {
        FreeLibrary(hSti);
    }

    return hr;
}
```