Canon

Canon EOS Digital SDK

EDSDK1.3 API Programming Reference

Revision 1.0
9/14/2006

Camera Design Dept.53
Camera Development Div.5
Camera Development Center

Revision History

Revision No.	Version	Date	Revised page(s)	Reason and content of revision	Reviser
1	1.0	9/14/2006		First release	

Table of Contents

Revision History/Date		Corrections	Reviser	Remarks

Revision History/Date		Corrections	Reviser	Remarks

Revision History/Date		Corrections	Reviser	Remarks

Revision History/Date		Corrections	Reviser	Remarks

1. Introduction

EDSDK stands for EOS Digital Camera Software Development Kit. EDSDK provides the functions required to control cameras connected to a host PC, digital images created in digital cameras, and images downloaded to the PC. This document describes the collection of functions implemented in the EDSDK library.

EDSDK provides an interface for accessing image data shot using a Canon EOS digital camera. Using EDSDK allows users to implement the following types of representative functions in software.

- Allows transfer of images in a camera to storage media on a host PC.
- Allows RAW images to be processed and saved in JPEG format.
- Allows remotely connected cameras and the image being shot to be controlled from a host PC.

1.1 Basic Topics

EDSDK provides a C language interface for accessing Canon EOS digital cameras and data created these cameras. EDSDK is designed to provide standard methods of accessing different camera models and their data. Using EDSDK allows users to implement Canon EOS digital camera features in software.

There are two versions of EDSDK. One runs under a Windows environment, while the other runs under a Macintosh environment.

Revision History/Date		Corrections	Reviser	Remarks

1.2 Supported Environments

EDSDK can be used on system configurations such as shown in the table below.

1.2.1 Development Environment

Windows	
OS	Windows 2000, XP (Home/Professional)
Memory	128 MB or more (256 MB or more when using XP)
Hard disk	50 MB or more available storage
Interface	USB2.0 or IEEE1394
Development environment	Microsoft Visual Studio6.0, Microsoft Visual Studio.NET2003 or later
Macintosh	
OS	Mac OSX 10.3.9-10.4 (10.4.7 or later on Intel-based Macintosh)
Memory	256 MB or more
Hard disk	50 MB or more available storage
Interface	USB2.0 or IEEE1394
Development environment	Xcode2.2 or later

1.2.2 Target Environment

Windows	
OS	Windows 2000, XP (Home / Professional), Vista
Memory	128 MB or more (256 MB or more when using XP)
Hard disk	50 MB or more available storage
Interface	USB2.0 or IEEE1394
Macintosh	
OS	Mac OSX 10.3.9-10.4 (10.4.7 or later on Intel-based Macintosh)
Memory	256 MB or more
Hard disk	50 MB or more available storage
Interface	USB2.0 or IEEE1394

1.3 Supported Cameras

Supports models beginning from the EOS 1D MarkII.
The following models are supported as of March 2006.

EOS 1D Mark II
EOS 20D
EOS 1Ds Mark II
EOS Kiss Digital N/350D/REBEL XT
EOS 5D
EOS 1D Mark II N
EOS 30D
EOS Kiss Digital X/400D/REBEL XTi

1.4 Installing EDSDK

1.4.1 Including Header Files

The following files are required in order to use the EDSDK using C/C++ language.

Revision History/Date		Corrections	Reviser	Remarks

EDSDK.h, EDSDKTypes.h, EDSDKErrors.h

Windows:

Be sure to copy the three header files listed above into the header access folder of the development environment.

Be sure to add them to the application project workspace.

*Since these are C language header files, it is necessary to provide header files depending on the programming language.

Macintosh:

Be sure to include the three header files listed above.

1.4.2 Linking the Library

After header files are included, it is necessary to link the EDSDK library as described below.

Windows:

There are two methods of linking EDSDK: one where EDSDK.lib files are copied to the folder specified by a development environment library path and EDSDK.lib is specified as an import module, and another where EDSDK.dll is loaded by the LoadLibrary function.
When loading EDSDK.dll, get pointers to each EDSDK function using the GetProcAddress function and assign them to function pointer variables. When calling each EDSDK function, make the call via the function pointer variable obtained here.

Macintosh:

Add EDSDK.framework to Groups＆Files.

Revision History/Date		Corrections	Reviser	Remarks

1.4.3 Executing the EDSDK Client Application

Windows:
All DLLs are required in order to execute an EDSDK client application.

Notes: Do not copy the collection of EDSDK library files to the system folder or extension folder.

Macintosh:

Place EDSDK.framework in an application directory such as Contents/frameworks/.

It is also possible to load “EDSDK.framework” as a source file. The following code has been written as an Objective-C source.

```
-(id)init {
        // START to Load EDSDK.framework ------------------------------------------------------------------
        NSString *symName = @"EDSDK.framework" ;
        int         i;
        NSArray *array = [NSBundle allFrameworks];
        void        *symData = NULL;

        for (i = 0; symData == NULL && i < [array count]; i++) {
            NSBundle *framework = [array objectAtIndex:i];
            NSString *bundleID = [framework bundleIdentifier];
            if (bundleID) {
                CFBundleRef bundle = CFBundleGetBundleWithIdentifier((CFStringRef) bundleID);
                if (bundle) {
                    symData = CFBundleGetFunctionPointerForName(bundle, (CFStringRef) symName);
                }
            }
        }
        // END of Loading EDSDK.framework ------------------------------------------------------------------

        EdsError err = EDS_ERR_OK ;
        err = EdsInitializeSDK() ;
}
```

Notes: Do not copy the EDSDK framework file to the system folder.

Revision History/Date		Corrections	Reviser	Remarks

2. Overview

2.1 Protocol for Remote Connection

Two types of protocol are used by EOS Digital to connect to a host PC. EDSDK client applications can basically communicate with remotely connected cameras without any awareness of the difference between protocols.

2.1.1 Type 1 (Legacy Protocol)

Legacy protocol is an original protocol from Canon for connections between a host PC and camera. This protocol is incorporated into cameras up to EOS5D and in EOS (EOS1 series) cameras with an IEEE1394 interface. A special device driver for the connected camera must be installed on the host PC in order to connect using this protocol. Be sure to install this driver beforehand from the CD-ROM supplied with Canon cameras or by downloading from Canon's homepage. (The required driver is installed in EDSDK.framework under Macintosh environments.)

Cameras which use a Type 1 protocol as standard such as EOS 1DmarkII N are called "Type 1 protocol standard cameras" in this manual.

2.1.2 Type 2 (PTP)

PTP is an abbreviation of "Picture Transfer Protocol." PTP is a standard protocol used to transfer images to a PC. This protocol is incorporated in EOS digital cameras that include a USB interface starting with EOS Kiss Digital N (EOS 350D/REBEL XT). A device driver for each model is unnecessary when connecting to an OS that supports PTP. (However, a device driver for making PTP connections is required when using an OS which does not support PTP as standard such as Windows 2000. This driver can be obtained from the CD-ROM supplied with Canon cameras or by downloading from Canon's homepage.)

Type 1 protocol has been eliminated from cameras with a USB interface starting from EOS30D and Type 2 protocol is utilized as that standard.

Cameras that use Type 2 protocol as standard such as EOS30D are called "Type 2 protocol standard cameras" in this manual.

> EOS Kiss Digital N , 350D, REBELXT, and EOS 5D model cameras come shipped from the factory with communications set for [Print/PTP] but functions that support PC connections are limited. For example, capture-related features cannot be used. Since these cameras use [PC connection] (Type 1 protocol) as the standard for connecting to a PC, they are Type 1 protocol standard cameras.

Revision History/Date		Corrections	Reviser	Remarks

2.1.3 Support by Model

The following table shows the protocol which can be used by EDSDK for each model when controlling a remotely connected camera. Be sure to set the communication settings of the camera as follows.

	Type 1 Protocol Standard Cameras						**Type 2 Protocol Standard Cameras**
	Models	1DMarkII, 1DsMarkII, 1DMarkII N	20D		Kiss Digital N/ 350D/REBELXT, 5D		30D, Kiss Digital X/ 400D/REBEL XTi
	Interface	IEEE1394	USB2.0		USB2.0		USB2.0
	Camera communication settings	—	PC con-nection	Print/PTP	PC con-nection	Print/PTP	Print/PC
Retrieval of camera setup information		○	○	×	○	×	○
Retrieval of image data in the camera		○	○	×	○	×	○
Camera control (capture)		○	○	×	○	×	○

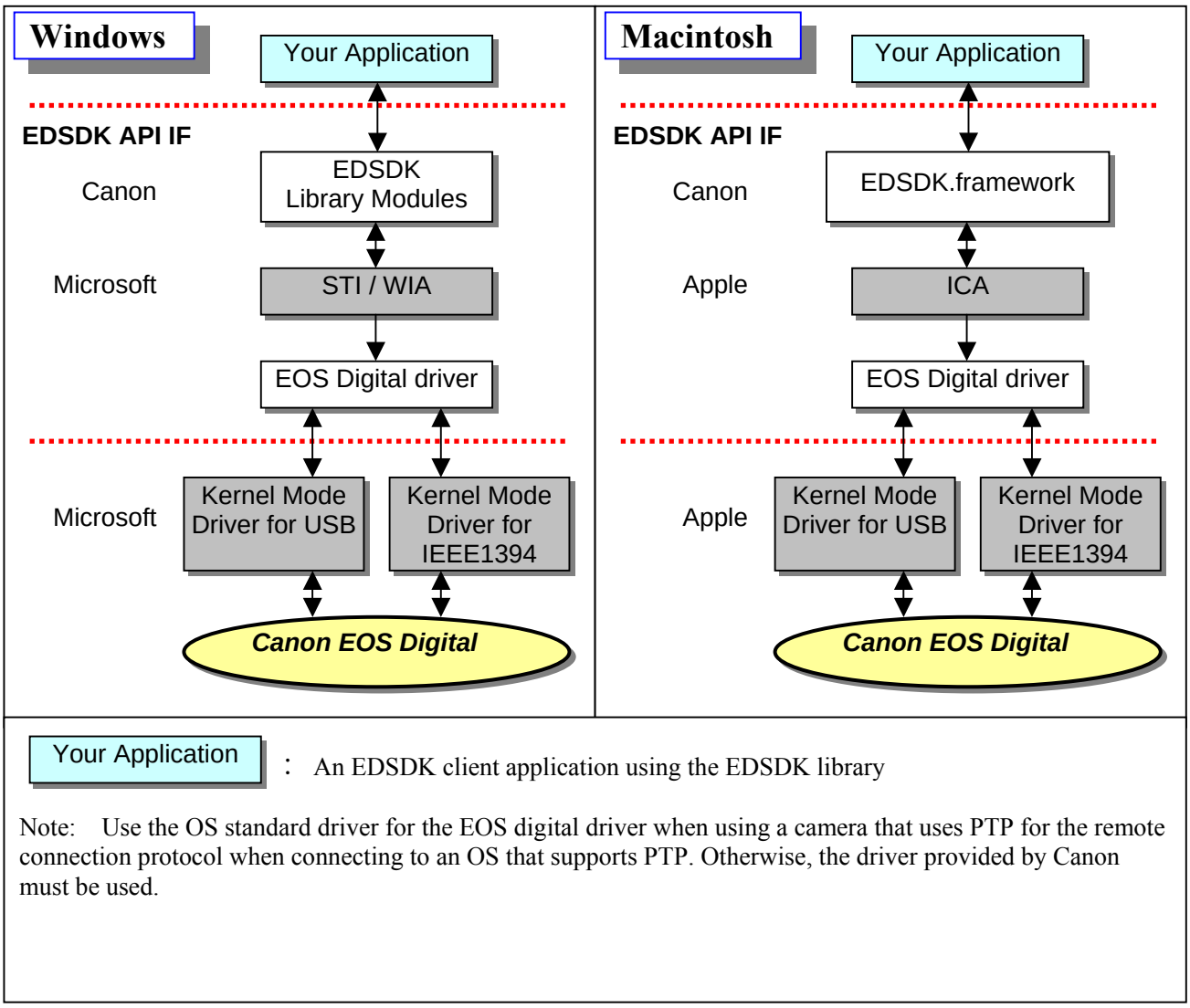
○：Available

×：Not available

Revision History/Date		Corrections	Reviser	Remarks

2.2 System Architecture

The following figure shows the configuration of software when an EOS digital camera has been connected.



Note: Use the OS standard driver for the EOS digital driver when using a camera that uses PTP for the remote connection protocol when connecting to an OS that supports PTP. Otherwise, the driver provided by Canon must be used.

Figure 2-1 System Architecture

Revision History/Date		Corrections	Reviser	Remarks

2.3 Library Modules

The following figure shows the module configuration of EDSDK.

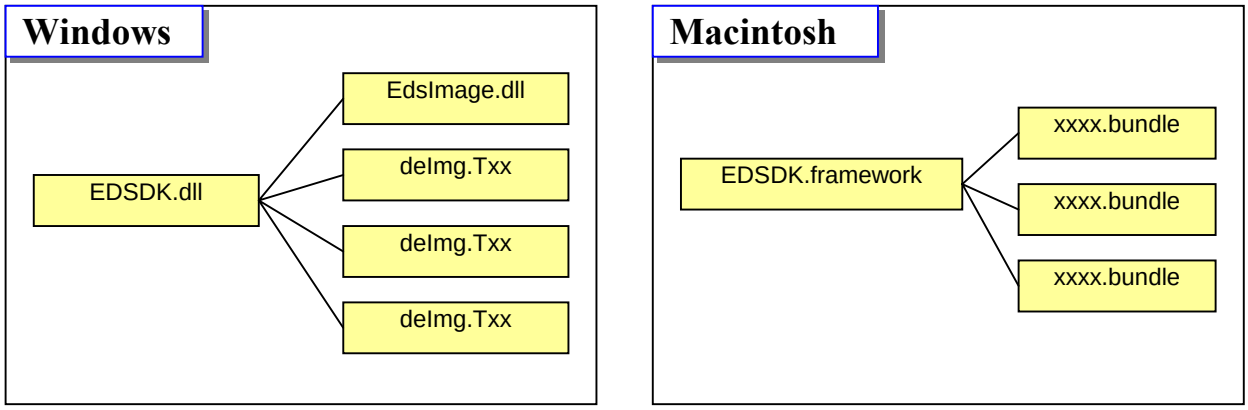


Figure 2-2 Library Module Configuration

Revision History/Date		Corrections	Reviser	Remarks

2.4 EDSDK Objects

As shown in Figure 1-3, EDSDK employs a hierarchical structure with a camera list at the root in order to control and access cameras connected to the host PC. This hierarchical structure consists of the following elements: camera list, cameras, volumes, folders, image files, audio files, etc.
These elements are treated as belonging to one of the following object categories: **EdsCameraListRef, EdsCameraRef, EdsVolumeRef,** and **EdsDirectoryItemRef.** Having a hierarchical structure, these four objects may have child objects.

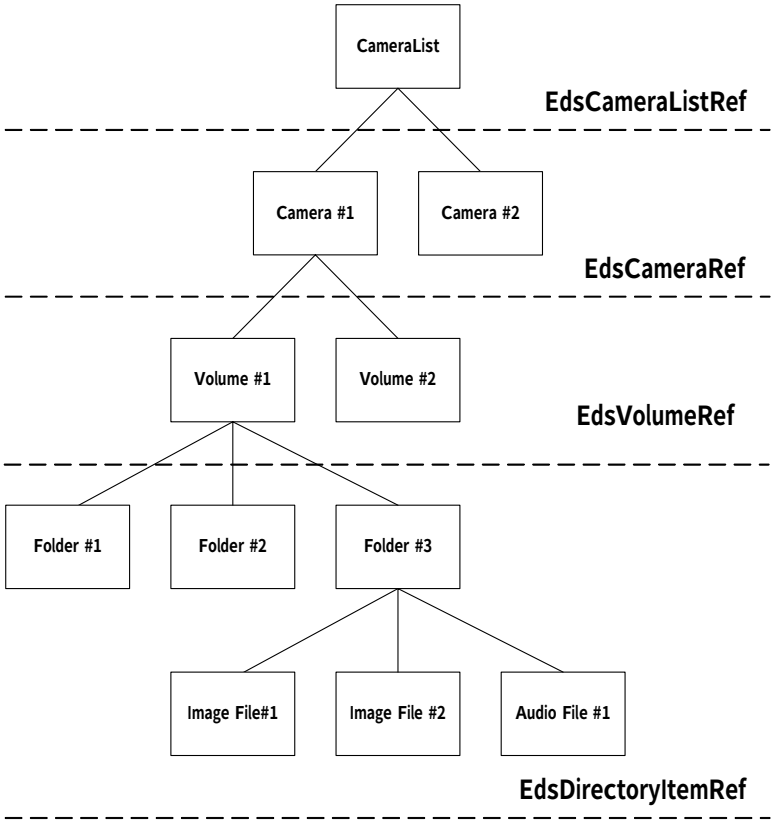


Figure 2-3 Hierarchical Structure of EDSDK Objects

Although the four objects shown above are used to access connected cameras, on an image file is transferred to the host PC, the object used to control that image changes even if it is the same image file.

As shown in Figure 1-4 below, the EdsStreamRef object is used to control input/output when transferring images from the camera to the host. Then EdsImageRef is used to control the image file transferred to the host. This is due to the fact that operations differ for an image file is stored in the camera versus an image file stored on the host.

Revision History/Date		Corrections	Reviser	Remarks

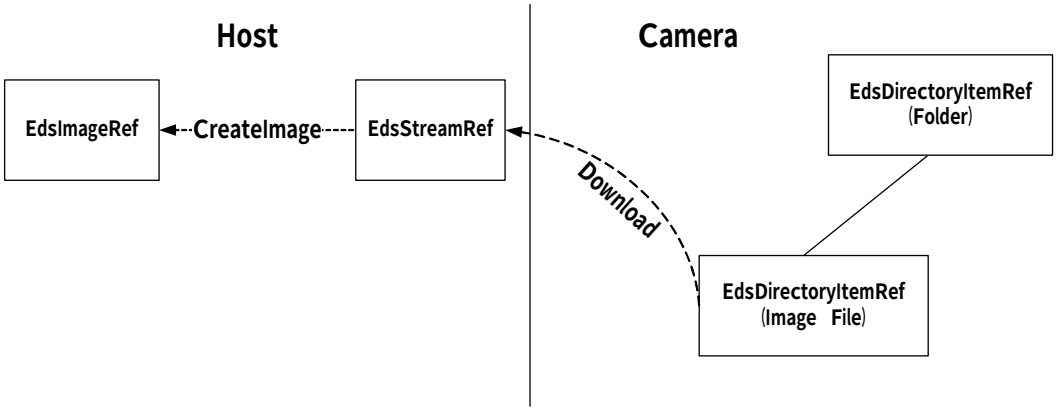


Figure 2-4 Changes in Controlled Objects

Bringing together the above information, the following objects can be handled using the EDSDK.

(1) **EdsCameraListRef**
This object represents an enumeration of the cameras remotely connected to the host PC by IEEE1394 or USB interface. This object can be used to select the camera to be controlled from among the cameras currently connected with EDSDK client application. This object can also be used when getting an EdsCameraRef child object.

(2) **EdsCameraRef**
This object represents a remotely connected camera. This object is used to control the camera or to get an EdsVolumeRef object when accessing the memory card, which is a child object of the camera.

(3) **EdsVolumeRef**
This object represents the memory card inside the camera. If the camera model allows two memory cards to be installed at once, as with the EOS1 line of cameras, the EdsVolumeRef object represents one memory card each. This object is used to get an EdsDirectoryItemRef object, which is a child object, when performing operations on a file or folder on the memory card.

(4) **EdsDirectoryItemRef**
This object represents a file or folder on the camera. When files are downloaded from the camera, each file to be downloaded is treated as one of these objects.

(5) **EdsImageRef**
This object represents image data. This data is obtained from image files. This object is used to retrieve and control information included with an image such as thumbnails and parameters.

(6) **EdsStreamRef**
This object represents the file I/O stream. An open stream on the host PC can be specified as the download destination when downloading files in the camera to the host PC. Streams are also used when loading image files stored on the storage media of the host PC into an EDSDK client application. Furthermore, EdsStreamRef objects can also be created in memory.

Revision History/Date		Corrections	Reviser	Remarks

2.5 Object Management

2.5.1 Object Management Using a Reference Counter

Applications built using the EDSDK carry out object management using a reference counter.

EDSDK stores a reference counter for all objects. The reference counter is set to 1 when an object has been allocated. The developer increases the reference counter by 1 at the point that the object is required by the program, and lowers it by 1 when the object is no longer needed. When a reference counter reaches 0, the associated object is automatically deleted by the EDSDK. The developer must, therefore, explicitly declare that an object is being referred when it is required by the program. EdsRetain and EdsRelease are provided as APIs for controlling object reference counters.

2.5.2 Releasing Resources when Exiting the Library

Applications built using the EDSDK will release all allocated resources when EdsTerminateSDK is called.

Revision History/Date		Corrections	Reviser	Remarks

2.6 Properties

Properties are stored under EDSDK for camera and image objects. For example, properties may represent values such as camera Av and Tv. The functions **EdsGetPropertyData** and **EdsSetPropertyData** are used to get and set these properties. Since this API takes objects of undefined type as arguments, the properties that can be retrieved or set differ depending on the given object. In addition, some properties have a list of currently settable values. **EdsGetPropertyDesc** is used to get this list of settable values. For details on types of properties, the objects the are associated with, and the role they play, see Properties.

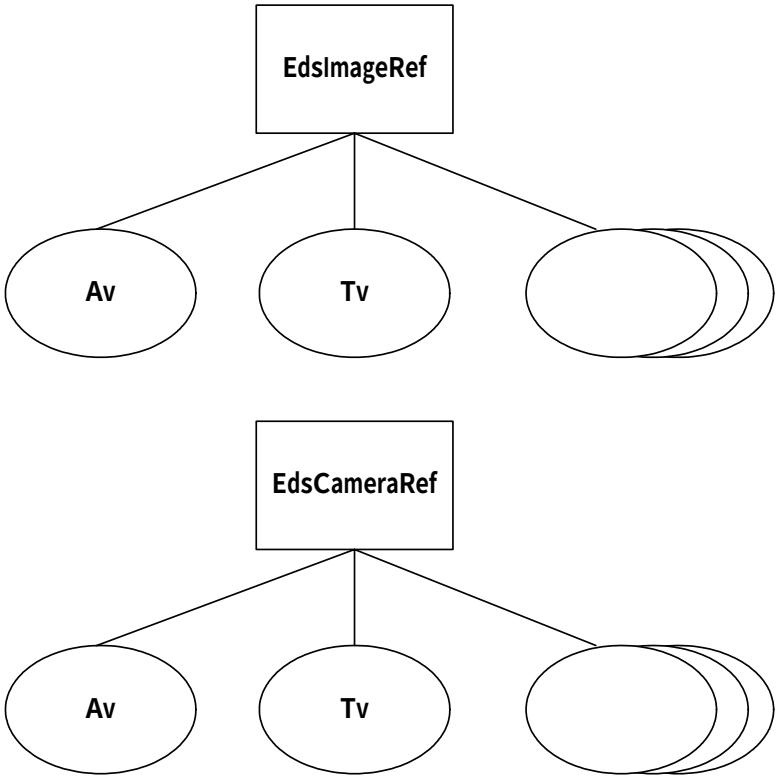


Figure 2-5 Example of Object Properties

Revision History/Date		Corrections	Reviser	Remarks

2.7 Camera Status

Cameras remotely connected to the host PC can be in one of several states: UI lock, UI lock release, direct transfer, and direct transfer release. Camera state transitions are shown in the figure below.

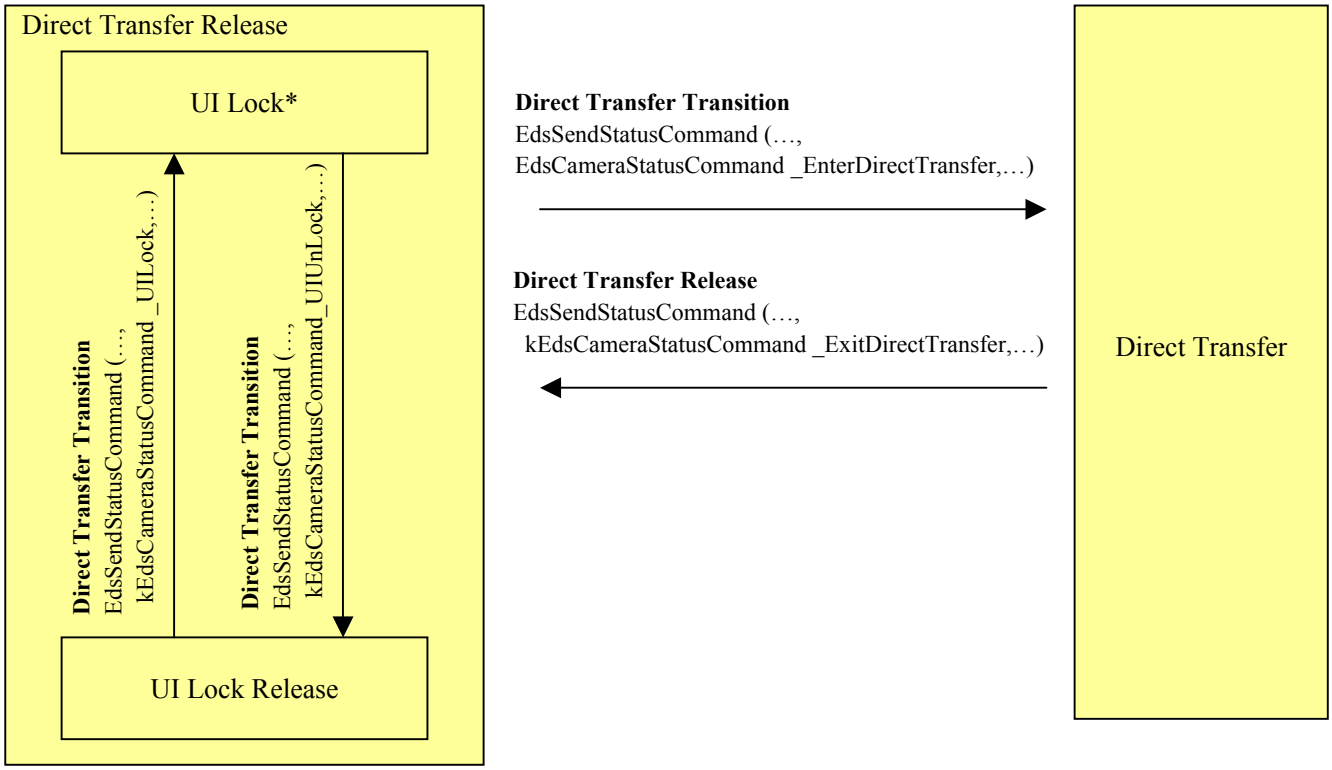
(1) UI Lock
In this state, all operations of the camera unit are disabled and only operations from the host PC are accepted.
This allows data and instructions to be safely sent from the host PC to the camera.

(2) UI Lock Release
In this state, operations of the camera unit are enabled. Although data and instructions can be sent from the host PC to the camera in this state, conflicts may arise.

(3) Direct Transfer (for models such as the EOS30D with an Easy Direct button)
In this state, the camera is currently directly transferring data. Available camera operations are limited to those functions related to the direct transfer. It is possible to send instructions from the PC to the camera in this state.
A direct transfer request event notification (kEdsObjectEvent_DirItemRequestTransferDT) is issued to the EDSDK client application connected to the camera when an operation for starting image download is initiated using camera controls. The EDSDK client application receives this event and begins processing for downloading images from the camera.

(4) Direct Transfer Release
This state indicates that direct transfer is not currently being carried out.

Revision History/Date		Corrections	Reviser	Remarks



* The camera sometimes automatically locks/releases when in the UI Lock state.

Figure 2-6 Camera State Transitions

Revision History/Date		Corrections	Reviser	Remarks

2.8 Asynchronous Events

An asynchronous event is a mechanism used to issue notifications from the EDSDK to the application regarding cameras connected to the host PC or state changes that have occurred for a camera. For example, if a state change occurs where a camera's shooting mode changes and a new image that needs to be transferred to the PC has been shot, a notification of that fact is sent to the application regardless of its state (asynchronously). An event handler capable of the specific processing required for a particular event must be registered in order to receive such an event (notification). An event handler is a user function called when an event is received. Event handlers are also referred to as "callback functions." Users can allow events to be accepted by creating and registering callback functions that accept events issued by EDSDK.

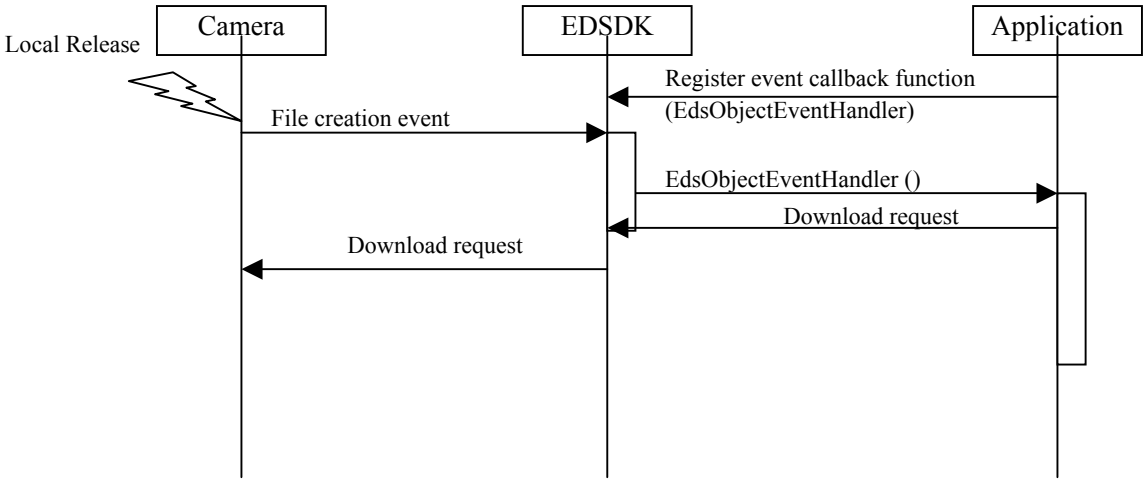


Figure 2-7 Example of a Camera Operation-Based Event Notification

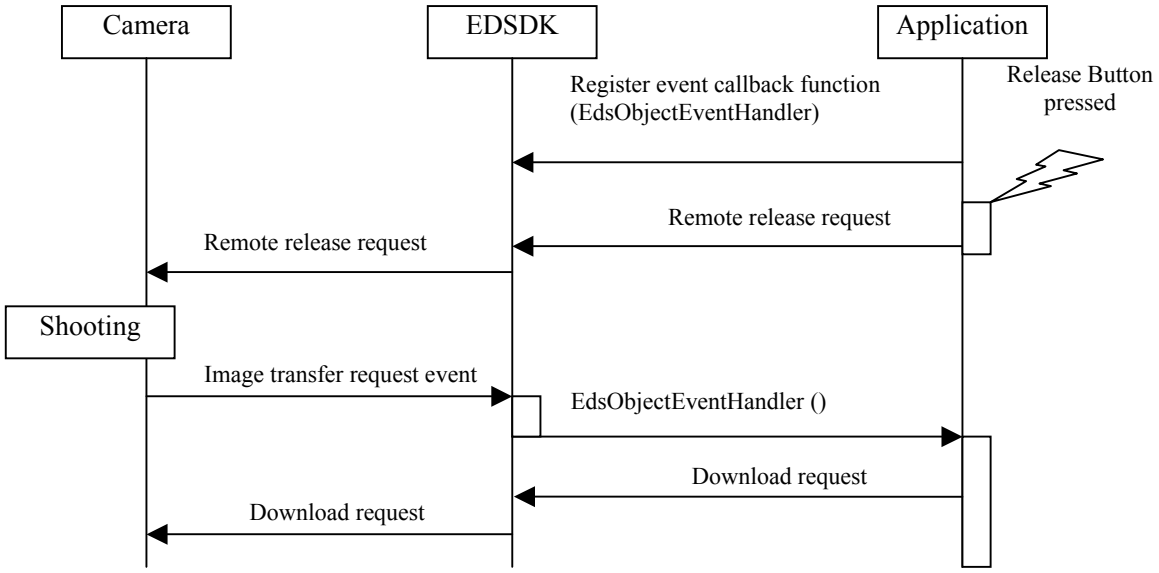


Figure 2-8 Host PC Operation-Related Event Notification

Revision History/Date		Corrections	Reviser	Remarks

When an event occurs, the EDSDK executes the callback function registered by the user. The callback function is executed on a newly generated thread and takes information depending on the event type as arguments (as specified by the event ID).
The user must release objects as they become unneeded.

There are three types of events issued from the EDSDK to a client application: object-related events, property-related events, and state-related events.

(1) Object-related events
This is the group of events where request notifications are issued to create, delete or transfer image data stored in a remotely connected camera (in memory) or image files on the memory card.
(2) Property-related events
This is the group of events where notifications are issued regarding changes in the properties of a remotely connected camera.
(3) State-related events
This is the group of events where notifications are issued regarding changes in the state of a remotely connected camera, such as the activation of a shut-down timer.

For details on event information and the role events play, see the section Asynchronous Events.

Revision History/Date		Corrections	Reviser	Remarks

2.9 Initializing and Terminating the Library

The user must initialize the EDSDK library in order to use EDSDK functions other than those for getting device information from a camera. The user must also terminate the library when EDSDK functions are no longer needed.

Be sure to execute initialization and termination of the library once each within the application process.

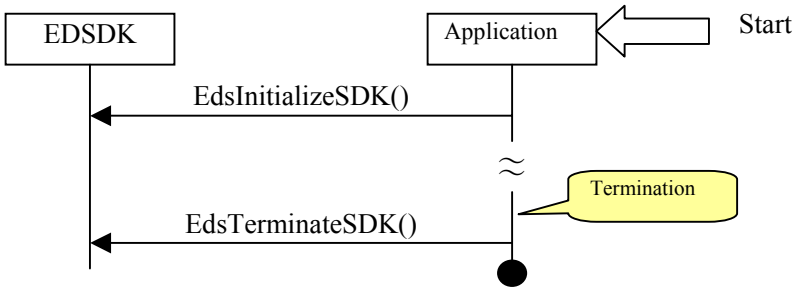


Figure 2-9 Initialization and Termination

Revision History/Date		Corrections	Reviser	Remarks

2.10 Accessing a Camera

The EDSDK provides methods of accessing and controlling a camera. In order to allow more than one camera connected to the host PC by USB or other means, it is possible to get all camera objects by repeatedly calling **EdsGetChildAtIndex** by specifying an index of child objects on the camera list.

The number of cameras connected can be obtained using **EdsGetChildCount**. Specify 0 as the index passed to **EdsGetChildAtIndex** if there is only one camera.

EDSDK client application can open a session with any one of the connected cameras. Opening a session means connecting to a camera at the application level so that it is possible to control that camera from the application and get associated properties and events. To open a session, specify the camera in question and call **EdsOpenSession.** Open sessions must be closed using **EdsCloseSession** when communications are finished.

Note that EDSDK does not support opening sessions with more than one camera at once.

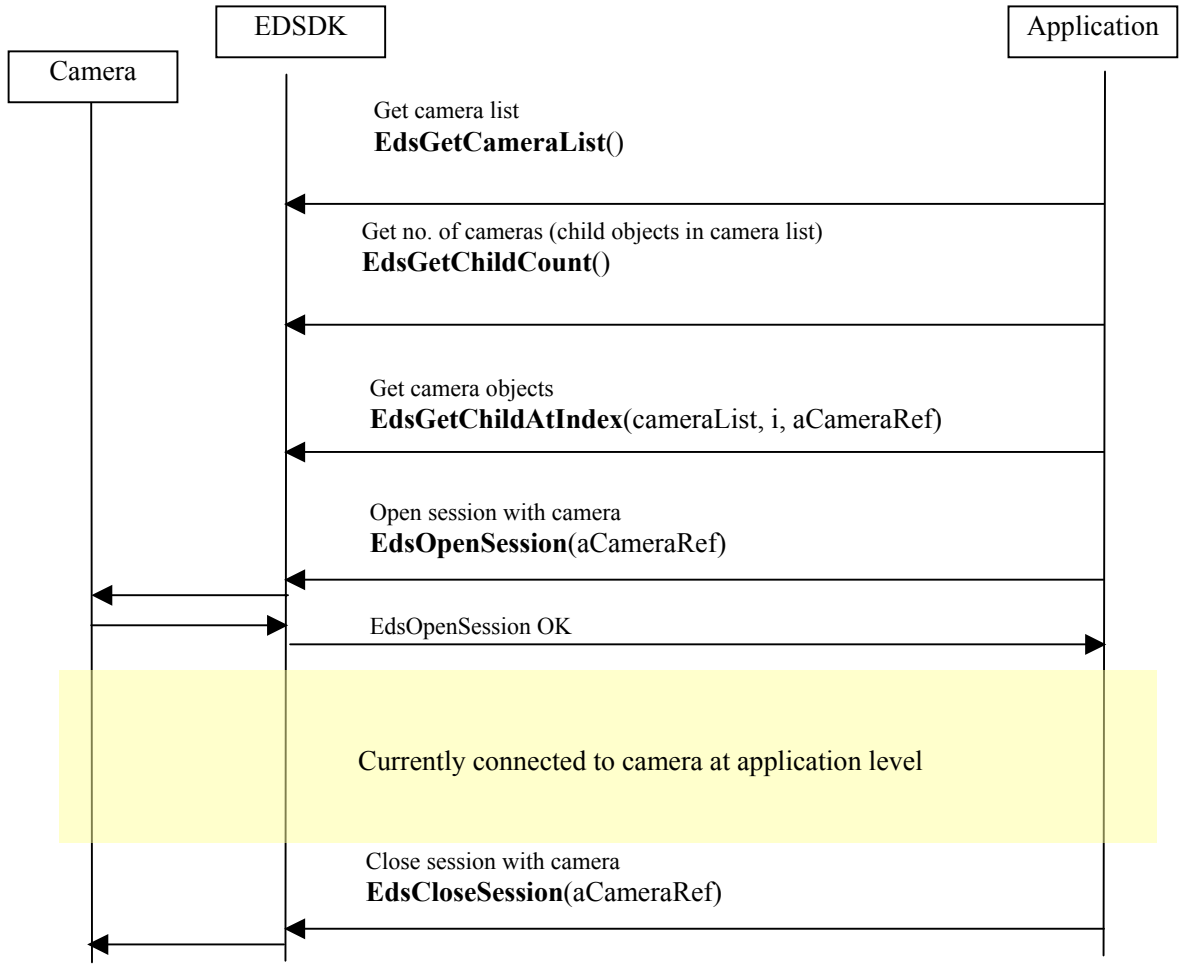


Figure 2-10 Camera Access

Revision History/Date		Corrections	Reviser	Remarks

Notes on Developing Windows Applications

When creating applications that run under Windows, a COM initialization is required for each thread in order to access a camera from a thread other than the main thread.

To create a user thread and access the camera from that thread, be sure to execute **CoInitializeEx(NULL, COINIT_APARTMENTTHREADED)** at the start of the thread and **CoUnInitialize()** at the end.

Sample code is shown below. This is the same when controlling EdsVolumeRef or EdsDirectoryItemRef objects from another thread, not just with EdsCameraRef .

```
void TakePicture(EdsCameraRef camera)
{
        // Executed by another thread
        HANDLE    hThread = (HANDLE)_beginthread(threadProc, 0, camera);
        // Block until finished
        ::WaitForSingleObject( hThread, INFINITE );
}


void threadProc(void* lParam)
{
        EdsCameraRef camera = (EdsCameraRef)lParam;

        CoInitializeEx( NULL, COINIT_APARTMENTTHREADED );

        EdsSendCommand(camera, kEdsCameraCommand_TakePicture, 0);

        CoUninitialize();

        _endthread();
}
```

Revision History/Date		Corrections	Reviser	Remarks

2.11 Transferring Files in the Camera

This section describes how to access files in the camera and transfer them to the host PC.
Although it is possible to access the camera and control the properties of files (such as the date of creation and protection settings), it is not possible to analize file properties. Files must therefore be transferred in order to get file properties. A method for transferring thumbnails (header information) only is also provided for such cases.

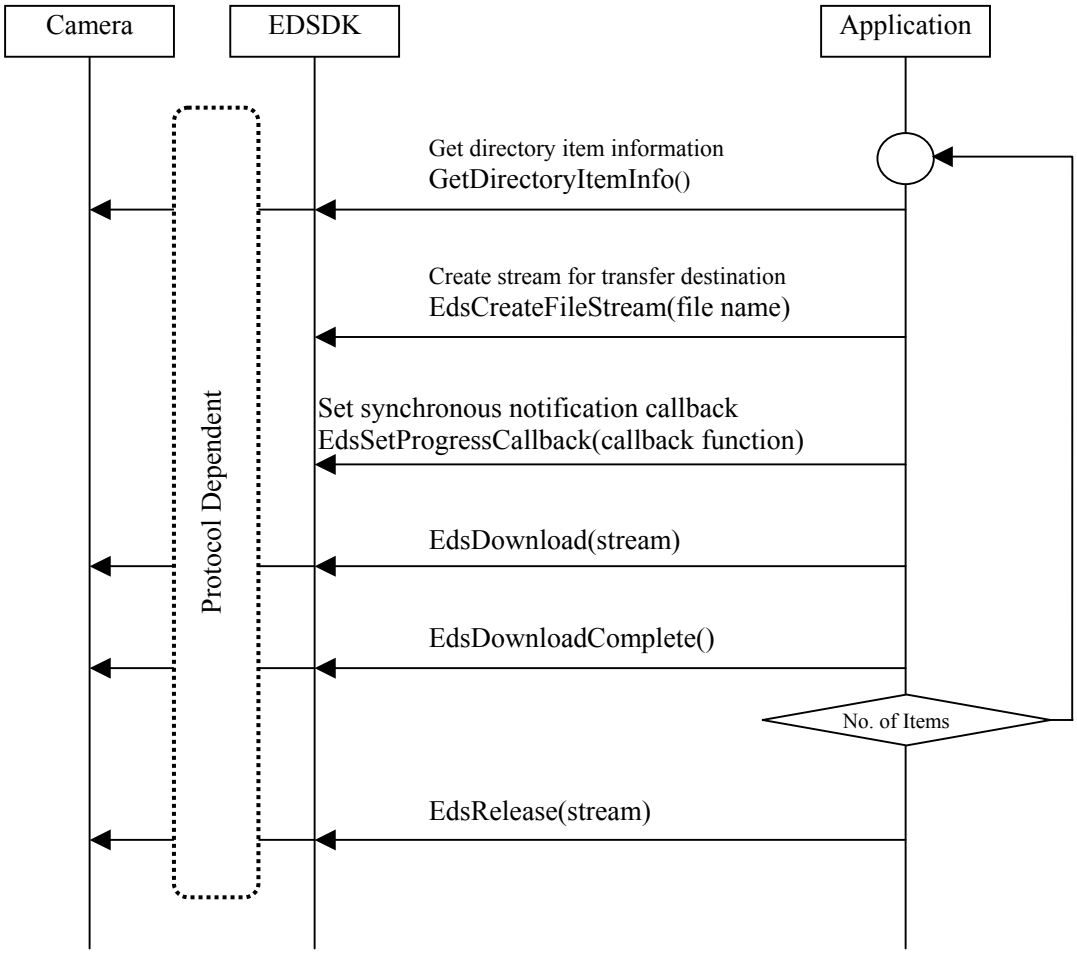


Figure 2-11 Transfer of Files in Camera

Revision History/Date		Corrections	Reviser	Remarks

2.12 Transferring Captured Images

When a shoot command is sent from the host PC to the camera, the camera will record the image shot in a buffer inside the camera. Once the shot has been taken, the callback function set using **EdsSetPropertyEventHandler, EdsSetObjectEventHandler,** and **EdsSetCameraStateEventHandler** will be called by the EDSDK. The user must sequentially transfer the images stored in the camera buffer to the host PC.

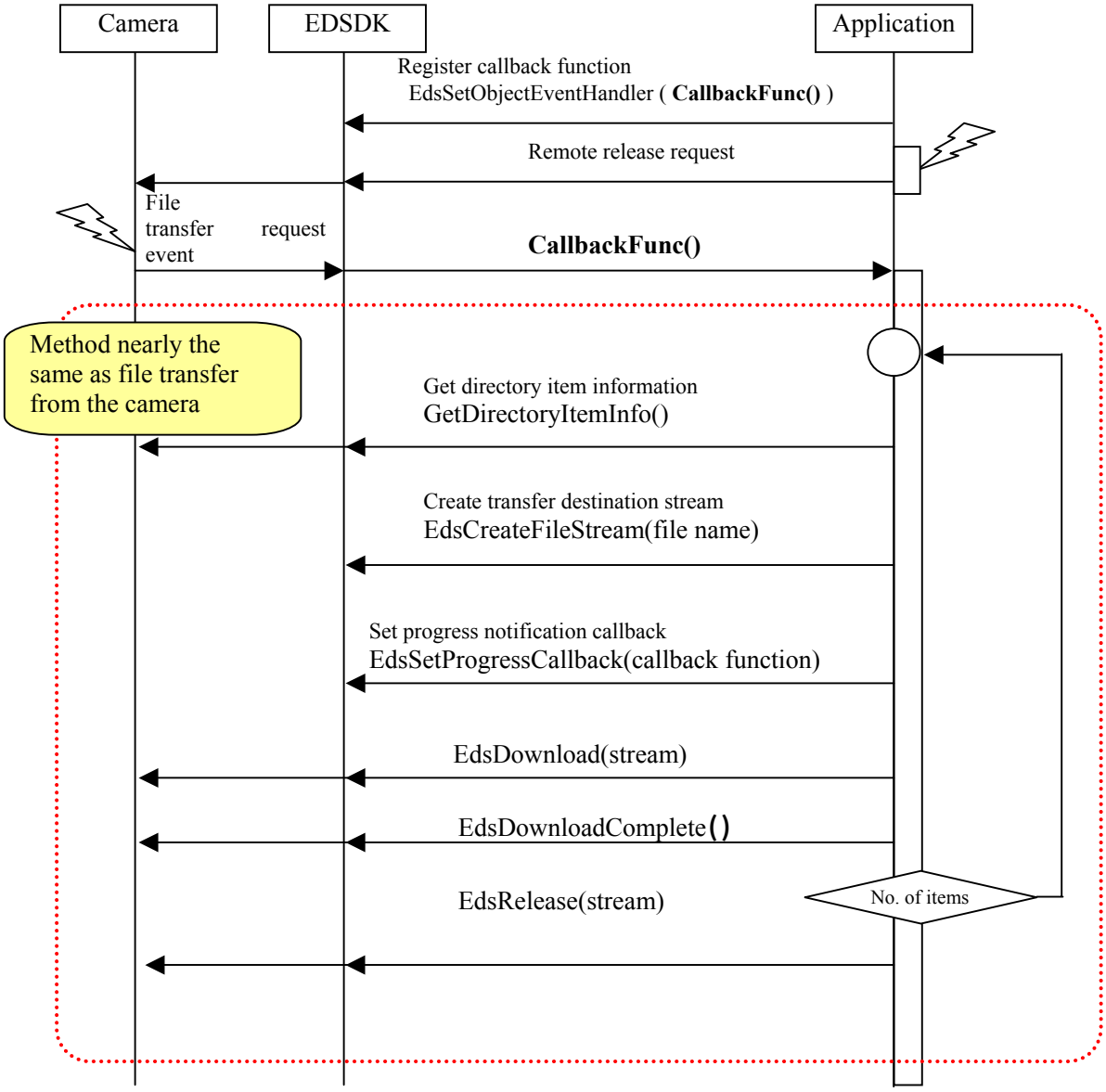


Figure 2-12 Capture Image Transfer

Revision History/Date		Corrections	Reviser	Remarks

2.13 Handling Image Objects

2.13.1 Overview

As touched on in the section on EDSDK objects, it is impossible to get an image object reference from an image file stored in a camera. An image object reference can only be obtained after first downloading the image file to a host PC.

An image object is an object that has properties. Camera properties such as Tv and Av that are used while shooting images are stored and can be obtained using **EdsGetPropertyData**. In addition, it is possible to process an image under conditions other than those at the time the image was shot by setting processing-related properties such as the white balance and picture style using **EdsSetPropertyData** if the image object is RAW.

2.13.2 Getting and Setting Properties

The following figure shows the sequence for getting properties from a camera image.

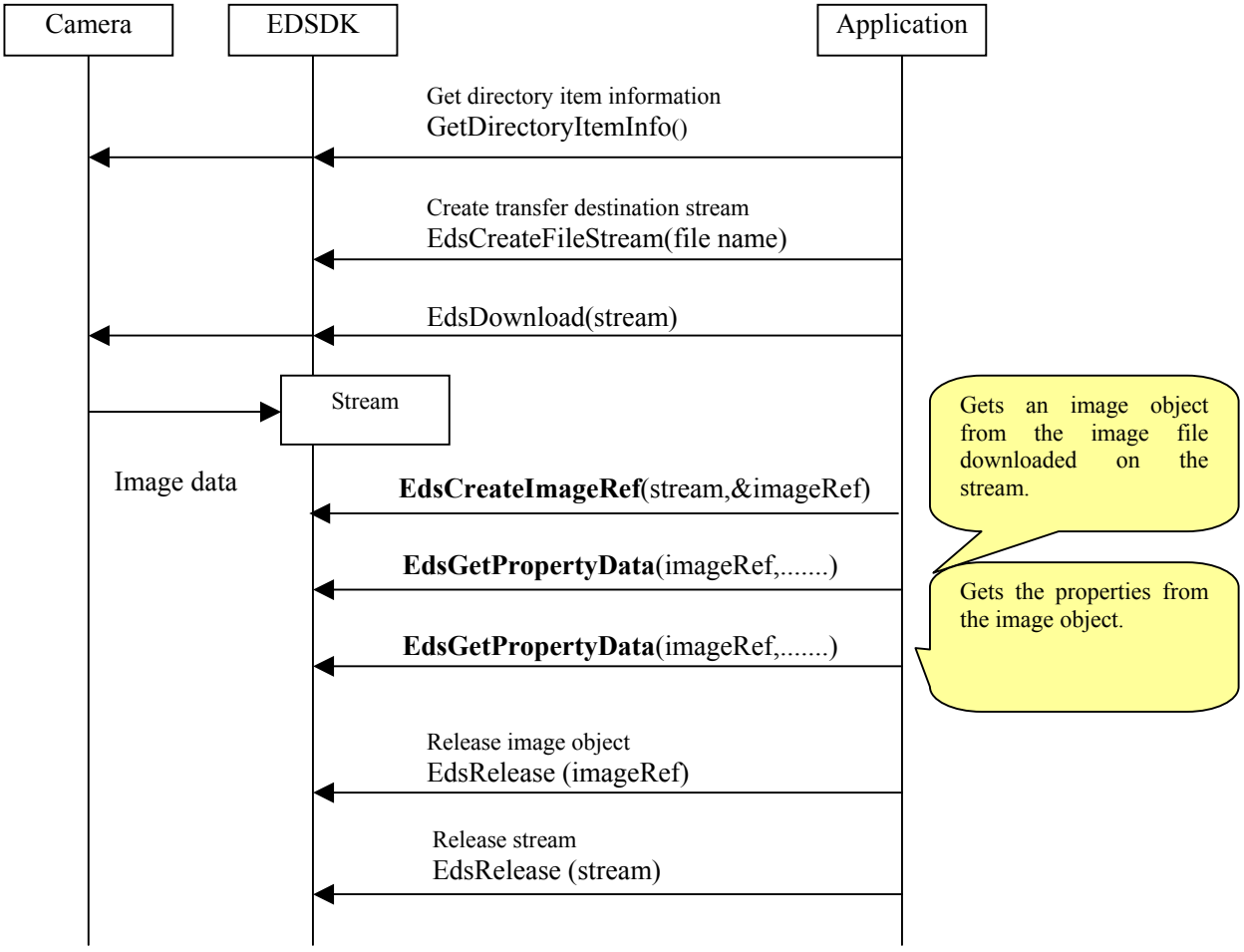


Figure 2-13 Getting an Image Object and Its Properties

Revision History/Date		Corrections	Reviser	Remarks

When processing is carried out using **EdsGetImage** or **EdsSaveImage** by setting properties for the image object, the specified property settings will be reflected in the generated JPEG. Note, however, that changes to properties will not be reflected in the source image stored by EdsImageRef.

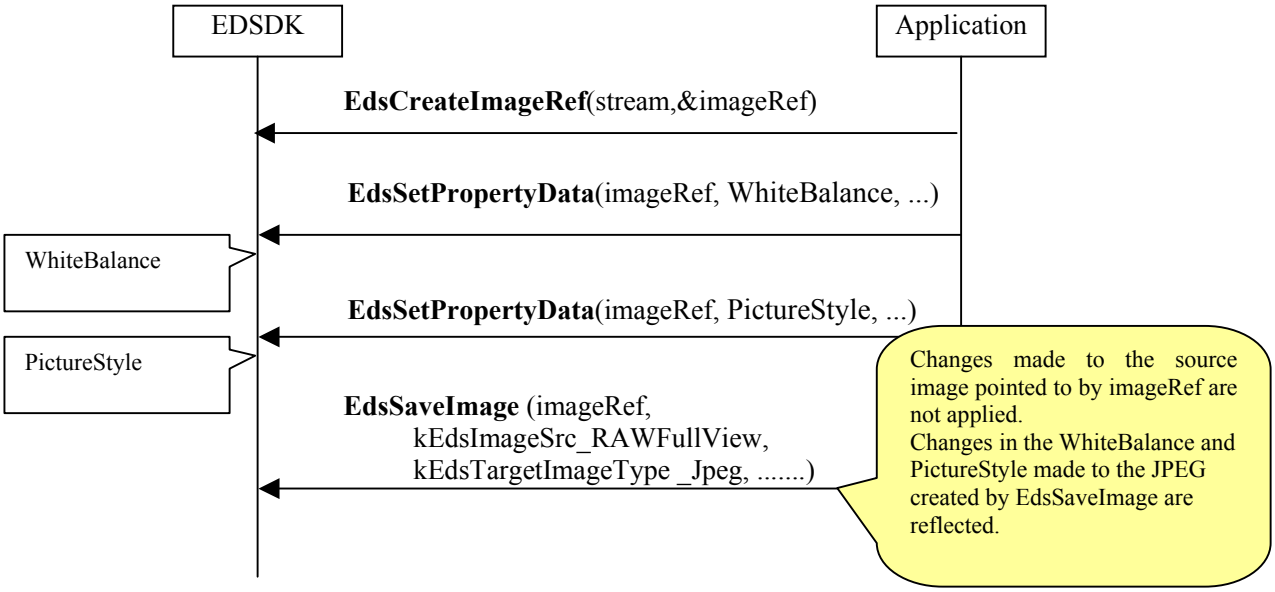


Figure 2-14 Setting Properties Reflected in the Resulting Processed Image

To save changes made to the source image RAW, first set properties and then execute **EdsReflectImageProperty**.

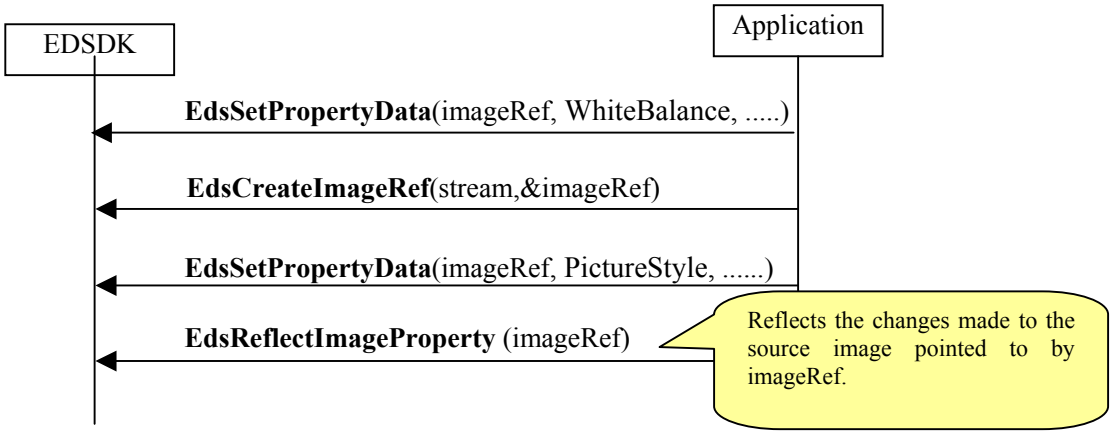


Figure 2-15 Saving Specified Properties for a Source Image

Revision History/Date		Corrections	Reviser	Remarks

Note:

Image objects are not thread-safe objects. Thus, when the same image object will be accessed by multiple threads, ensure exclusive control in your application.

Revision History/Date		Corrections	Reviser	Remarks

2.14 Basic Data Type Definitions

This section introduces the basic data types used under the EDSDK. These data types are defined as C language types.

```
typedef void                    EdsVoid;
typedef int                     EdsBool;

typedef char                    EdsChar;
typedef char                    EdsInt8;
typedef unsigned char           EdsUInt8;
typedef short                   EdsInt16;
typedef unsigned short          EdsUInt16;
typedef long                    EdsInt32;
typedef unsigned long           EdsUInt32;

#ifdef __MACOS__
#ifdef __cplusplus
    typedef long long           EdsInt64;
    typedef unsigned long long  EdsUInt64;
#else
    typedef SInt64              EdsInt64;
    typedef UInt64              EdsUInt64;
#endif
#else
    typedef __int64             EdsInt64;
    typedef unsigned __int64    EdsUInt64;
#endif

typedef float                   EdsFloat;
typedef double                  EdsDouble;
```

2.15 EDSDK Errors

Most of the APIs supplied by EDSDK return an error code of type EdsError as their return value.

The return value of an API that terminates normally is EDS_ERR_OK. If an error occurs, the return value of the API in question is set to the error code indicating the root cause of the error and any passed parameters are stored as undefined values. (Note that an API used to control files is not limited to returning an error related to file control.)

For error codes, see the list given in the header file EdsError.h or see EDS ERROR Lists at the end of the section describing APIs in this document.

Revision History/Date		Corrections	Reviser	Remarks

3. API Reference

3.1 API Details

API specifications are explained in the following format.

Description

Indicates the main API function.

Syntax

EdsError EdsXXXXX(EdsUInt32 **in**XXXX, EdsBaseRef *__out__XXX) ;

Indicates the syntax for calling the API.

Parameters

Explains each argument in the syntax individually.
In the syntax, argument names in the format **in**XXX represent arguments for which you enter values. Argument names in the format **out**XXX represent arguments with values set by the libraries (that is, passed by reference). Before calling APIs, you must prepare variables for storing the data to be retrieved.

Return Values

Explains API return values.

See Also

Indicates information related to the API.

Note

Considerations when using the API.

Example

Sample code.

Revision History/Date		Corrections	Reviser	Remarks

3.1.1 EdsInitializeSDK

Description

Initializes the libraries.
When using the EDSDK libraries, you must call this API once before using EDSDK APIs.

Syntax

EdsError EdsInitializeSDK( )

Parameters

None

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs
EdsTerminateSDK

Example

• See Sample 1.

3.1.2 EdsTerminateSDK

Description

Terminates use of the libraries.
Calling this function releases all resources allocated by the libraries.

Syntax

EdsError EdsTerminateSDK()

Parameters

None

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs
EdsInitializeSDK

Example

• See Sample 1.

3.1.3 EdsRetain

Description

Increments the reference counter of existing objects.

Syntax

EdsUInt32 EdsRetain(EdsBaseRef inRef)

Revision History/Date		Corrections	Reviser	Remarks

Parameters

inRef

Objects of all types in the EDSDK can be designated.

Type	Description
EdsCameraListRef	A list of remote cameras
EdsCameraRef	A particular remote camera
EdsVolumeRef	A volume on the camera's recording media
EdsDirectoryItemRef	A directory or file in the volume
EdsImageRef	An image file on the host computer
EdsStreamRef	Stream data on the remote camera or host computer

Return Values

Returns a reference counter if successful. For errors, returns 0xFFFFFFFF.
The return value is 4 bytes, and the maximum value of the reference counter is 65535.

See Also

• Related APIs
EdsRelease

Example

• See Sample 1.

3.1.4 EdsRelease

Description

Decrements the reference counter to an object. When the reference counter reaches 0, the object is released.

Syntax

EdsUInt32 EdsRelease (EdsBaseRef inRef)

Parameters

inRef

Objects of all types in the EDSDK can be designated.
(EdsCameraListRef, EdsCameraRef, EdsDirectoryItemRef, EdsImageRef, or EdsStreamRef)

Return Values

Returns a reference counter if successful. For errors, returns 0xFFFFFFFF.

See Also

• Related APIs
EdsRetain, EdsGetCameraList, EdsGetChildAtIndex, and EdsGetParent, EdsCreateImage

Note

• The reference counter is incremented not only for objects with a reference counter incremented explicitly by means of EdsRetain but also for EDSDK objects retrieved by means of EdsGetCameraList, EdsGetChildAtIndex, or EdsGetParent (refer to the objects that can be designated with inRef), for which the reference counter is incremented by one implicitly. Thus, when objects are no longer needed, you must use this API to decrease the reference counter.

Example

Revision History/Date		Corrections	Reviser	Remarks

• See Sample 1.

3.1.5 EdsGetChildCount

Description

Gets the number of child objects of the designated object.
Example: Number of files in a directory

Syntax

EdsError EdsGetChildCount (EdsBaseRef inRef, EdsUInt32* outCount)

Parameters

inRef

EdsCameraListRef, EdsVolumeRef, EdsCameraRef, or EdsDirectoryItemRef.

outCount

Pointer to the variable for receiving the child object of the object designated by inRef.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs
EdsGetChildAtIndex

Example

• See Sample 2.

3.1.6 EdsGetChildAtIndex

Description

Gets an indexed child object of the designated object.

Relevant object	Child object that can be retrieved
Camera list	Camera
Camera	Volume
Volume	Directory item
Directory item	Directory item (folder or file)

Syntax

```
EdsError  EdsGetChildAtIndex(
                EdsBaseRef      inRef,
                EdsInt32        inIndex,
                EdsBaseRef*     outRef      )
```

Parameters

inRef

Designate the parent object of the object to get. You can designate EdsCameraListRef, EdsCameraRef, EdsVolumeRef, or EdsDirectoryItemRef.

inIndex

Designate the index of the child object list. The index is 0-based, so designate 0 to get the first child object.

Revision History/Date		Corrections	Reviser	Remarks

outRef
The indexed child object.

Return Values
Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also
• Related APIs
EdsGetChildCount and EdsGetParent

Note
The reference counter is implicitly 1 for the retrieved child object. When the object is not needed, you must use EdsRelease to decrease the reference counter.

Example
• See Sample 2.

3.1.7 EdsGetParent

Description
Gets the parent object of the designated object.

Syntax
EdsError EDSAPI EdsGetParent(EdsBaseRef inRef, EdsBaseRef *outParentRef);

Parameters
inRef
The EdsCameraListRef, EdsCameraRef, EdsVolumeRef, or EdsDirectoryItemRef object.
outParentRef
Returns a pointer to the variable for receiving the parent object reference.

Return Values
Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also
• For details on object parent-child relationships, see EDSDK Objects.
• Related APIs
EdsGetChildAtIndex and EdsRelease

Note
The reference counter is implicitly 1 for the retrieved parent object. When the object is not needed, you must use EdsRelease to decrease the reference counter.

3.1.8 EdsGetCameraList

Description
Gets camera list objects.

Syntax
EdsError EdsGetCameraList(EdsCameraListRef *outCameraListRef)

Revision History/Date		Corrections	Reviser	Remarks

Parameters

outCameraListRef

When the return value is EDS_ERR_OK, a list of cameras connected to the host computer is specified in outCameraListRef.
When the return value is other than EDS_ERR_OK, the content of outCameraListRef is unspecified.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs
EdsRelease, EdsGetChildCount, and EdsGetChildAtIndex

Note

• The reference counter is implicitly 1 for the retrieved camera list. When the object is not needed, you must use EdsRelease to decrease the reference counter.

Example

• See Sample 2.

3.1.9 EdsGetDeviceInfo

Description

Gets device information, such as the device name.
Because device information of remote cameras is stored on the host computer, you can use this API before the camera object initiates communication (that is, before a session is opened).

Syntax

```
EdsError    EdsGetDeviceInfo(
EdsCameraRef    inCameraRef,
EdsDeviceInfo    *outDeviceInfo )
```

Parameters

inCameraRef

The camera object for which to get device information.

outDeviceInfo

Pointer to the EdsDeviceInfo structure for receiving device information.

EdsDeviceInfo

EdsDeviceInfo constituent elements	Type	Description
szPortName	EdsChar[]	Port name
szDeviceDescription	EdsChar[]	Device name Example: "EOS 30D PTP"
deviceSubType	EdsUInt32	Canon legacy protocol cameras: 0 Canon PTP cameras: 1

If the camera involved in PTP communication is connected to a Windows computer on which WIA is installed, 0 is specified in DeviceSubType, representing standard Windows PTP.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

Revision History/Date		Corrections	Reviser	Remarks

3.1.10 EdsGetVolumeInfo

Description

Gets volume information for a memory card in the camera.

Syntax

```
EdsError    EdsGetVolumeInfo(
                        EdsVolumeRef    inVolumeRef,
                        EdsVolumeInfo *outVolumeInfo )
```

Parameters

inVolumeRef

Designate the volume object for which to get volume information.

outVolumeInfo

Specifies the pointer to the EdsVolumeInfo structure for receiving the volume information.

EdsVolumeInfo

EdsVolumeInfo constituent elements	Type	Description
storageType	EdsUInt32	Value defined by Enum EdsStorageType
access	EdsAccess	Value defined by Enum EdsAccess
maxCapacity	EdsUInt64	Maximum size (in bytes)
freeSpaceInBytes	EdsUInt64	Available capacity (in bytes)
szVolumeLabel	EdsChar[]	Volume name (an ASCII string) Example: "A:" or another drive name

Enum EdsStorageType <defined location>EDSDKTypes.h

Value	Description
0	No memory card inserted
1	Compact flash
2	SD card

Enum EdsAccess <defined location>EDSDKTypes.h

Value	Description
0	Read Only
1	Write Only
2	Read and Write
0xFFFFFFFF	Access error Note: This means that the designated memory card is in a state preventing use, such as when the card is not formatted.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

- Related APIs
  EdsGetChildAtIndex

Note

- In the context of the EDSDK, volumes are objects representing memory cards.
- The constituent element access of EdsVolumeInfo is the access type when the file object is open.

Revision History/Date		Corrections	Reviser	Remarks

3.1.11 EdsGetDirectoryItemInfo

Description

Gets information about the directory or file objects on the memory card (volume) in a remote camera.

Syntax

```
EdsError    EdsGetDirectoryItemInfo(
                        EdsDirectoryItemRef    inDireItemRef,
                        EdsDirectoryItemInfo*    outDirItemInfo      )
```

Parameters

inDireItemRef

Designate the directory item object.

outDirItemInfo

Pointer to the DirectoryItemInfo structure for receiving the directory item information.
DirectoryItemInfo includes the following information.

Constituent elements	Description
size	The file size. For folders, the file size is indicated as 0.
isFolder	If a folder: True If not a folder: False
groupID	A non-zero integer. The same group ID is assigned to files that belong to the same group, such as RAW+JPEG images or RAW+AVI images. Note: Valid for type 2 protocol standard cameras.
option	An option when a direct transfer request is received (a kEdsObjectEvent_DirItemRequestTransferDT event). kEdsTransferOptionToDesktop is set when [Wallpaper] in the direct transfer is executed by means of camera operations. Prohibit it under other timing conditions. Note: Valid for type 2 protocol standard cameras.
szFileName	Returns the directory name or file name if successful. Example: "_MG_0060.JPG"

EdsTargetImageType <defined location>EDSDKTypes.h

Value	Description
kEdsTargetImageType_unknown	Folder, or unknown image type
kEdsTargetImageType _Jpeg	JPEG
kEdsTargetImageType _TIFF	8-bit TIFF
kEdsTargetImageType _TIFF16	16-bit TIFF
kEdsTargetImageType _RGB	8-bit RGB, chunky format
kEdsTargetImageType _RGB16	16-bit RGB, chunky format

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

Note

- For type 1 protocol standard cameras, you can determine if objects are in the same group by whether their file names (excluding the extension) of the szFileName member in the DirectoryItemInfo structure are the same or not.

Revision History/Date		Corrections	Reviser	Remarks

See Also

• For information on data types of the EDSDK, see "Data Types Used by the APIs" in the Appendix.

Example

• See Sample 6.

3.1.12 EdsOpenSession

Description

Establishes a logical connection with a remote camera.
Use this API after getting the camera's EdsCamera object.

Syntax

EdsError EDSAPI EdsOpenSession(EdsCameraRef inCameraRef);

Parameters

inCameraRef

Designate the camera object of the camera to connect to.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

Note

Use the EdsCloseSession API to disconnect from the camera.

See Also

• Related APIs
EdsCloseSession

Example

• See Sample 1.

3.1.13 EdsCloseSession

Description

Closes a logical connection with a remote camera.

Syntax

EdsError EDSAPI EdsCloseSession(EdsCameraRef inCameraRef);

Parameters

inCameraRef

Designate the camera object of the camera to disconnect from.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs
EdsOpenSession

Revision History/Date		Corrections	Reviser	Remarks

Example

• See Sample 1.

3.1.14 EdsSendCommand

Description

Sends a command such as "Shoot" to a remote camera.

Syntax

```
EdsError EdsSendCommand( EdsCameraRef    inCameraRef,
                         EdsUInt32    inCommand,    EdsUInt32    inParam )
```

Parameters

inCameraRef

Only a camera object can be designated.

inCommand

The command ID to send to the object.

In EDSDKTypes.h, you can designate commands defined by enum EdsCameraCommand.

inCommand	inParam	Description
kEdsCameraCommand_TakePicture	N/A	Requests the camera to shoot.
kEdsCameraCommand_ExtendShutDownTimer	N/A	Requests to extend the time for the auto shut-off timer. (Keep Device On)

inParam

Currently unused. Designate 0.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

Example

• See Sample 9.

3.1.15 EdsSendStatusCommand

Description

Sets the remote camera state or mode.

Syntax

```
EdsError EDSAPI EdsSendStatusCommand ( EdsCameraRef       inCameraRef,
                                       EdsCameraStatusCommand    inStatusCommand,
                                       EdsInt32                  inParam);
```

Parameters

inCameraRef

Designate the camera object.

inStatusCommand

Designate the particular mode ID to set the camera to.

In EDSTypes.h, you can designate commands defined by enum EdsCameraStatusCommand.

inStatusCommand	inParam	Description

Revision History/Date		Corrections	Reviser	Remarks

kEdsCameraStatusCommand _UILock	N/A	Locks the UI
kEdsCameraStatusCommand _UIUnLock	N/A	Unlocks the UI
kEdsCameraStatusCommand _EnterDirectTransfer	N/A	Puts the camera in direct transfer mode
kEdsCameraStatusCommand _ExitDirectTransfer	N/A	Ends direct transfer mode

inParam

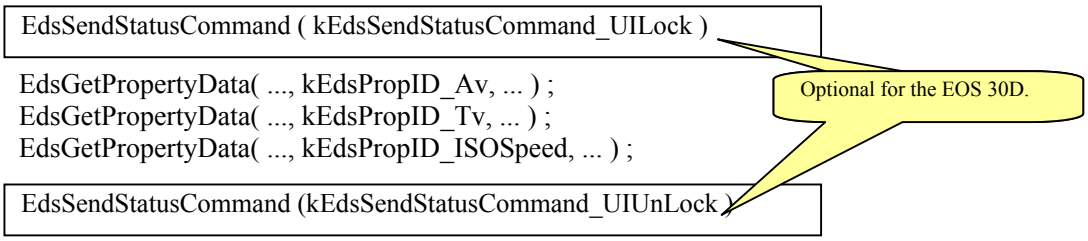
Currently unused. Designate 0.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

Note

- These are pairs of commands to lock and unlock the UI, as well as to put the camera in direct transfer mode and exit this mode. If you switch modes by means of EdsSendStatusCommand, use EdsSendStatusCommand again to restore the original mode.
- The UI must be locked on type 1 protocol standard cameras before sending a command to get or set the property. However, on type 2 protocol standard cameras, the UI is locked automatically by the camera, so locking the UI from the application is not necessary.



3.1.16 EdsGetPropertySize

Description

Gets the byte size and data type of a designated property from a camera object or image object.

Syntax

```
EdsError  EdsGetPropertySize( EdsBaseRef  inRef,
                              EdsPropertyID  inPropertyID,  EdsInt32  inParam,
                              EdsDataType  *outEdsDataType,  EdsUInt32  *outSize )
```

Parameters

inRef

Designate either EdsCameraRef or EdsImageRef.

inPropertyID

Designate the property ID.

inParam

Additional information of the property. Used to designate multiple additional items of information, if the property has such information that can be set or retrieved. For descriptions of values that can be designated for each property, see the description of inParam for EdsGetPropertyData.

outEdsDataType

Returns the property data type. The particular item defined by enum EdsDataType is returned.

outSize

Stores the property size. The data type and value returned varies depending on the property ID. See

Revision History/Date		Corrections	Reviser	Remarks

"Property Details" for further information.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

- Related APIs
  EdsGetPropertyData and EdsGetPropertyDesc
- For further information on properties, see Properties.

Example

See Sample 3.

3.1.17 EdsGetPropertyData

Description

Gets property information from the object designated in inRef.

Syntax

```
EdsError  EDSAPI EdsGetPropertyData(
                    EdsBaseRef      inRef,
                    EdsPropertyID   inPropertyID,
                    EdsInt32        inParam,
                    EdsUInt32       inPropertySize,
                    EdsVoid         *outPropertyData )
```

Parameters

inRef

Designate the object for which to get properties. The EDSDK objects you can designate are EdsCameraRef, EdsDirectoryItemRef, or EdsImageRef.

inPropertyID

Designate the property ID.

inParam

Designate additional property information. Use additional property information if multiple items of information such as picture styles can be set or retrieved for a property.
Values that can be designated for each property are as follows.

■ Properties regarding camera settings

InPropertyID	inParam setting value
kEdsPropID_ProductName	0
kEdsPropID_BodyID	0
kEdsPropID_OwnerName	0
kEdsPropID_MakerName	0
kEdsPropID_DateTime	0
kEdsPropID_FirmwareVersion	0
kEdsPropID_BatteryLevel	0
kEdsPropID_UserWhiteBalanceData	User-defined dataset number
kEdsPropID_UserToneCurveData	User-defined dataset number
kEdsPropID_UserPictureStyleData	Current Picture Style: 0 Picture Styles registered in User 1–3: kEdsPictureStyle_User1 to kEdsPictureStyle_User3

Revision History/Date		Corrections	Reviser	Remarks

kEdsPropID_CFn	Custom Function number
kEdsPropID_PFn	Personal Function number
kEdsPropID_SaveTo	0

■ **Properties regarding images**

InPropertyID	inParam setting value
kEdsPropID_ImageQuality	0
kEdsPropID_JpegQuality	(1) EOS 1D series models High-order Word: Processing Parameter set number; low-order Word: kEdsImageQualityNormal or kEdsImageQualityFine (2) Other models Image Size (retrieved by means of kEdsPropID_ImageQuality)
kEdsPropID_Orientation	0
kEdsPropID_ICCProfile	0
kEdsPropID_FocusInfo	0
kEdsPropID_WhiteBalance	0
kEdsPropID_ColorTemperature	0
kEdsPropID_WhiteBalanceShift	0
kEdsPropID_ClickWBPoint	0
kEdsPropID_WBCoeffs	0
kEdsPropID_Linear	0
kEdsPropID_Sharpness	To designate the current sharpness value (or, if EdsImageRef is designated, either the current value or the value at the time of shooting): 0 To designate the ParameterSet number by designating EdsCameraRef: the ParameterSet number
kEdsPropID_ParameterSet	0
kEdsPropID_ColorMatrix	0
kEdsPropID_ColorSaturation	To designate the current saturation value (or, if EdsImageRef is designated, either the current value or the value at the time of shooting): 0 To designate ColorMatrix by designating EdsCameraRef: one of the ColorMatrix numbers
kEdsPropID_Contrast	Current contrast value (or, if EdsImageRef is designated, either the current value or the value at the time of shooting): 0 To designate the ParameterSet number by designating EdsCameraRef: the ParameterSet number
kEdsPropID_ColorTone	Current color tone value (or, if EdsImageRef is designated, either the current value or the value at the time of shooting): 0 To designate ColorMatrix by designating EdsCameraRef: one of the ColorMatrix numbers
kEdsPropID_ColorSpace	Current color space value (or, if EdsImageRef is designated, either the current value or the value at the time of shooting): 0 To designate ColorMatrix by designating EdsCameraRef: one of the ColorMatrix numbers To designate a picture style by designating EdsCameraRef: one of enum EdsPictureStyle
kEdsPropID_PhotoEffect	0
kEdsPropID_FilterEffect	Current filter effect value (or, if EdsImageRef is designated, either the current value or the value at the time of shooting): 0
kEdsPropID_ToningEffect	Current toning effect value (or, if EdsImageRef is designated, either the current value or the value at the time of shooting): 0

Revision History/Date		Corrections	Reviser	Remarks

kEdsPropID_ToneCurve	Standard (read-only; cannot be set): 0 Set 1:1 Set 2:2 Set 3:3 and so on Note: If EdsImageRef is designated, only 0.
kEdsPropID_PictureStyle	Current picture style value (or, if EdsImageRef is designated, either the current value or the value at the time of shooting): 0 One of these: User setting 1: kEdsPictureStyle_User1 User setting 2: kEdsPictureStyle_User2 User setting 3: kEdsPictureStyle_User3
kEdsPropID_PictureStyleCaption	0

■ Properties regarding image capture

InPropertyID	inParam setting value
kEdsPropID_AEMode	0
kEdsPropID_DriveMode	0
kEdsPropID_ISOSpeed	0
kEdsPropID_MeteringMode	0
kEdsPropID_AFMode	0
kEdsPropID_Av	0
kEdsPropID_Tv	0
kEdsPropID_ExposureCompensation	0
kEdsPropID_DigitalExposure	0
kEdsPropID_FlashCompensation	0
kEdsPropID_FocalLength	0
kEdsPropID_AvailableShots	0
kEdsPropID_Bracket	0
kEdsPropID_WhiteBalanceBracket	0
kEdsPropID_LensName	0
kEdsPropID_AEBracket	0
kEdsPropID_FEBracket	0
kEdsPropID_ISOBracket	0
kEdsPropID_NoiseReduction	0
kEdsPropID_FlashOn	0
kEdsPropID_RedEye	0
kEdsPropID_FlashMode	0

inPropertySize

Designate the byte size of the property. If the property data size is not known in advance, it can be retrieved by means of EdsGetPropertySize.

outPropertyData

Specifies the property data. The data type and value returned vary depending on the property. For property information, see Properties.

Return Values

Returns EDS_ERR_OK on normal completion. Otherwise, see the EDS Error Lists for error codes.

See Also

- Related APIs
  EdsGetPropertySize, EdsSetPropertyData, and EdsGetPropertyDesc
- For further information on properties, see Properties.

Revision History/Date		Corrections	Reviser	Remarks

Note

Regarding retrieval of the camera property data in particular, the conditions that can be retrieved vary depending on the values of other property data. For further information, see Properties.

Example

- See Sample 3.

3.1.18 EdsSetPropertyData

Description

Sets property data for the object designated in inRef.

Syntax

```
EdsError   EdsSetPropertyData (
                EdsBaseRef        inRef,
                EdsPropertyID     inPropertyID,
                EdsInt32          inParam,
                EdsUInt32         inPropertySize,
                const EdsVoid*    inPropertyData    )
```

Parameters

inRef

Designate the object for which to set properties. Designate either EdsCameraRef or EdsImageRef.

inPropertyID

Designate the property ID.

inParam

Designate additional property information. Use additional property information if multiple items of information such as picture styles can be set or retrieved for a property. For descriptions of values that can be designated for each property, see the description of inParam for EdsGetPropertyData.

inPropertySize

Designate the size of the property data in bytes. The data size of each property can be retrieved by means of EdsGetPropertySize.

inPropertyData

Designate the property data to set.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

- Related APIs
  EdsGetPropertySize, EdsGetPropertyData, EdsGetPropertyDesc, and EdsReflectImageProperty
- For further information on properties, see Properties.

Note

- When you set properties of an image object (EdsImageRef), this API maintains the change internally. Execute EdsReflectImageProperty and incorporate the updated image property in the file (stream).
- When setting properties in type 1 protocol standard cameras, take steps to prevent contention with

Revision History/Date		Corrections	Reviser	Remarks

camera operations, such as by locking the UI. On the other hand, for type 2 protocol standard cameras, the UI can be locked or unlocked on the camera itself, so do not lock the UI.

Example

• See Sample 5.

3.1.19 EdsGetPropertyDesc

Description

Gets a list of property data that can be set for the object designated in inRef, as well as maximum and minimum values.
This API is intended for only some shooting-related properties.

Retrievable properties for settable data lists	Description
kEdsPropID_AEMode	Shooting mode
kEdsPropID_ISOSpeed	ISO speed
kEdsPropID_MeteringMode	Metering mode
kEdsPropID_Av	Aperture value
kEdsPropID_Tv	Shutter speed
kEdsPropID_ExposureCompensation	Exposure compensation

Syntax

```
EdsError EdsGetProperyDesc(
                EdsBaseRef        inRef,
                EdsPropertyID     inPropertyID,
                EdsPropertyDesc*  outProperyDesc )
```

Parameters

inRef

The target object. Designate EdsCameraRef, EdsDirectoryItemRef, or EdsImageRef.

inPropertyID

Designate a property ID.

outProperyDesc

Specifies a pointer to the EdsPropertyDesc structure for getting a list of property data that can currently be set in the target object.
If the API return value is EDS_ERR_OK, a settable property data list of properties that can be set is specified, as retrieved from the target object.

The structure of the list of property data that can be set (**EdsPropertyDesc**) has the following constituent elements.

EdsPropertyDesc constituent elements	Type	Description
form	EdsInt32	Reserved (currently, always 0)
access	EdsAccess	Reserved (currently, always 0)
numElements	EdsInt32	Indicates the number of property data list elements stored in the PropDesc array.
propDesc	EdsInt32[]	A property data array. The meaning of PropDesc array elements varies depending on the property type.

Revision History/Date		Corrections	Reviser	Remarks

Return Values

EDS_ERR_INVALID_PARAMETER is returned if a property ID is designated in inPropertyID that cannot be used with GetPropertyDesc.
Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

- Related APIs
  EdsGetPropertySize, EdsGetPropertyData, EdsSetPropertyData, and EdsGetPropertyDesc
- For details on properties and the meaning of array elements that can be set in the data list, see the Properties section.
- For information on data types of the EDSDK, see "Data Types Used by the APIs" in the Appendix.

Example

- See Sample 4.

3.1.20 EdsDeleteDirectoryItem

Description

Deletes a camera folder or file.
If folders with subdirectories are designated, all files are deleted except protected files.
EdsDirectoryItem objects deleted by means of this API are implicitly released by the EDSDK. Thus, there is no need to release them by means of EdsRelease.

Syntax

EdsError EDSAPI EdsDeleteDirectoryItem(EdsDirectoryItemRef inDirItemRef)

Parameters

inDirItemRef

Designate the folder or file to delete.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

- Related APIs
  EdsSendCommand

Note

- Be careful when deleting files on the remote camera to avoid doing so when the camera is not in the right mode. Lock the UI, for example.

3.1.21 EdsFormatVolume

Description

Formats volumes of memory cards in a camera.

Syntax

EdsError EDSAPI EdsFormatVolume (EdsVolumeRef inVolumeRef)

Revision History/Date		Corrections	Reviser	Remarks

Parameters

inVolumeRef

Designate the volume (memory card) to format.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs

EdsGetVolumeInfo

Note

• Be careful to avoid doing this when the camera is not in the right mode. Lock the UI, for example.

3.1.22 EdsGetAttribute

Description

Gets attributes of files on a camera.

Syntax

```
EdsError EDSAPI EdsGetAttribute ( EdsDirectoryItemRef    inDirItemRef,
                                  EdsFileAttributes      *outFileAttribute ) ;
```

Parameters

inDirItemRef

Designate the file object for which to get attributes.

outFileAttribute

Indicates the file attributes.

As for the file attributes, OR values of the value defined by enum EdsFileAttributes can be retrieved. Thus, when determining the file attributes, you must check if an attribute flag is set for target attributes.

Example: Determining the attribute value fileAttr, retrieved from a file object

```
if (kEdsFileAttribute_ReadOnly & fileAttr ){
    // The file is read-only
}
```

Enum EdsFileAttribtes <defined location>EDSDKTypes.h

Value	Description
kEdsFileAttribute_Normal	A standard file
kEdsFileAttribute_ReadOnly	Read-only
kEdsFileAttribute_Hidden	Hidden attribute
kEdsFileAttribute_System	System attribute
kEdsFileAttribute_Archive	Archive attribute

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs

EdsSetAttribute

Revision History/Date		Corrections	Reviser	Remarks

3.1.23 EdsSetAttribute

Description

Changes attributes of files on a camera.

Syntax

```
EdsError EDSAPI EdsSetAttribute ( EdsDirectoryItemRef   inDirItemRef,
                                  EdsFileAttributes     inFileAttribute ) ;
```

Parameters

inDirItemRef

Designate the file object for which to change attributes.

outFileAttribute

Indicates the file attributes.

As for the file attributes, OR values of the value defined by enum EdsFileAttributes can be retrieved.

Enum EdsFileAttribtes <defined location>EDSDKTypes.h

Value	Description
kEdsFileAttribute_Normal	A standard file
kEdsFileAttribute_ReadOnly	Read-only
kEdsFileAttribute_Hidden	Hidden attribute
kEdsFileAttribute_System	System attribute
kEdsFileAttribute_Archive	Archive attribute

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs
EdGetAttribute

3.1.24 EdsDownload

Description

Downloads a file on a remote camera (in the camera memory or on a memory card) to the host computer. The downloaded file is sent directly to a file stream created in advance.
When dividing the file being retrieved, call this API repeatedly. Also in this case, make the data block size a multiple of 512 (bytes), excluding the final block.

Syntax

```
EdsError    EDSAPI    EdsDownload(
EdsDirectoryItemRef       inDirItemRef,
EdsUInt32                 inReadSize,
EdsStreamRef              outStreamRef      )
```

Parameters

inDirItemRef

Designate the file object in the camera to download.

inReadSize

Designate the size in bytes to download.

outStreamRef

Specifies the destination stream. The stream for downloading is created by means of

Revision History/Date		Corrections	Reviser	Remarks

EdsCreateFileStream, EdsCreateMemoryStream, or the like.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs
EdsDownloadComplete, EdsDownloadCancel, EdsDownloadThumbnail, EdsCreateFileStream, EdsCreateMemoryStream, and EdsSetProgressCallback

Note

• EdsDownload is an API that may be checked with a progress callback. Using EdsSetProgressCallback to register the callback function enables the progress to be retrieved as an event during file transfer.
• Immediately after this API is called, the EdsDownloadComplete API must be called to notify the camera that the file transfer is complete. Similarly, if the download is canceled, EdsDownloadCancel must be called.
• If this API abends, a communication error between the camera and host computer occurs. If so, release the resources allocated by the application and restore the initial mode.

Example

• See Sample 6.

3.1.25 EdsDownloadComplete

Description

Must be called when downloading of directory items is complete. Executing this API makes the camera recognize that file transmission is complete.
This operation need not be executed when using EdsDownloadThumbnail.

Syntax

EdsError EDSAPI EdsDownloadComplete(EdsDirectoryItemRef inDirItemRef)

Parameters

inDirItemRef

Designate the file for which to complete the downloading process.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs
EdsDownload and EdsDownloadCancel

Note

• If transfer of a file that was divided is canceled, call EdsDownloadCancel instead of this API to notify the camera that downloading of the directory item has been canceled.

Example

• See Sample 6.

Revision History/Date		Corrections	Reviser	Remarks

3.1.26 EdsDownloadCancel

Description

Must be executed when downloading of a directory item is canceled. Calling this API makes the camera cancel file transmission. It also releases resources.
This operation need not be executed when using EdsDownloadThumbnail.

Syntax

EdsError EDSAPI EdsDownloadCancel (EdsDirectoryItemRef inDirItemRef)

Parameters

inDirItemRef

Designate the file for which to cancel downloading.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs
EdsDownload and EdsDownloadComplete

Note

• In applications that take locally released images on the camera and load them on host computer, if the application receives a file transfer request from the camera when the file is not needed (by means of kEdsObjectEvent_DirItemRequestTransfer or kEdsObjectEvent_DirItemRequestTransferDT), this API must be called to notify the camera that transmission has been canceled.
Normally, delete callback function registration at the moment an event is not needed.

3.1.27 EdsDownloadThumbnail

Description

Extracts and downloads thumbnail information from image files in a camera.
Thumbnail information in the camera's image files is downloaded to the host computer. Downloaded thumbnails are sent directly to a file stream created in advance.

Syntax

EdsError EDSAPI EdsDownloadThumbnail(
EdsDirectoryItemRef inDirItemRef,
EdsStreamRef outStreamRef)

Parameters

inDirItemRef

Designate the image file object with thumbnails to extract.

outStreamRef

Designate the stream for saving extracted thumbnails.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs
EdsDownload, EdsCreateFileStream, EdsCreateFileStreamEx, EdsCreateImageRef, and EdsGetImageInfo

Revision History/Date		Corrections	Reviser	Remarks

3.1.28 EdsCreateFileStream

Description

Creates a new file on a host computer (or opens an existing file) and creates a file stream for access to the file.
If a new file is designated before executing this API, the file is actually created following the timing of writing by means of EdsWrite or the like with respect to an open stream.

Syntax

EdsError EdsCreateFileStream (const EdsChar* inFileName,
EdsFileCreateDisposition inCreateDisposition,
EdsAccess inDesiredAccess, EdsStreamRef* outStream)

Parameters

inFileName

Designate the file name of a new file or a file to open.
You can designate a null-terminated string up to EDS_MAX_NAME characters long as the file name.

inCreateDisposition

Designate how the file is handled (that is, its disposition) if it exists or does not exist.
Designate a value defined in Enum EdsFileCreateDisposition.

Enum EdsFileCreateDisposition <defined location>EDSDKTypes.h

Value	Description
kEdsFileCreateDisposition_CreateNew	Creates a new file. An error occurs if the designated file already exists.
kEdsFileCreateDisposition_CreateAlways	Creates a new file. If the designated file already exists, that file is overwritten and existing attributes is erased.
kEdsFileCreateDisposition_OpenExisting	Opens a file. An error occurs if the designated file does not exist.
kEdsFileCreateDisposition_OpenAlways	If the file exists, it is opened. If the designated file does not exist, a new file is created.
kEdsFileCreateDisposition_TruncateExsisting	Opens a file and sets the file size to 0 bytes.

inDesiredAccess

Values defined in Enum EdsAccess may be designated.

Enum EdsAccess <defined location>EDSDKTypes.h

Value	Description
kEdsAccess_Read	Open a read-only stream.
kEdsAccess_Write	Open a write-only stream.
kEdsAccess_ReadWrite	Allow reading and writing.

outStreamRef

Returns a file stream to the open file.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

Revision History/Date		Corrections	Reviser	Remarks

See Also

- Related APIs
  EdsCreateFileStreamEx, EdsWrite, EdsRead, and EdsRelease

Note

- The maximum file name length is limited to EDS_MAX_NAME. To go beyond this limitation or enable support of Unicode file names, use the Unicode version, EdsCreateFileStreamEx.
- The stream you create must be released after use by means of EdsRelease.

Example

- See Sample 6.

3.1.29 EdsCreateFileStreamEx

Description

An extended version of EdsCreateFileStream.
Use this function when working with Unicode file names.

Syntax

```
EdsError  EdsCreateFileStreamEx(
#ifdef __MACOS__
    const    CFURLRef          inURL,
#else
    const    WCHAR*            inFileName,
#endif
    EdsFileCreateDisposition   inCreateDisposition,
    EdsAccess inDesiredAccess, EdsStreamRef* outStream)
```

Parameters

inURL (for Macintosh)
Designate CFURLRef.
inFileName (for Windows)
Designate the file name.
inDesiredAccess
See EdsCreateFileStream.
inCreateDisposition
See EdsCreateFileStream.
outStreamRef
Returns a file stream to the open file.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

- Related APIs
  EdsCreateFileStream, EdsWrite, EdsRead, and EdsRelease

Note

- This API is an extended version of EdsCreateStreamFromFile.
- The stream you create must be released after use by means of EdsRelease.

Revision History/Date		Corrections	Reviser	Remarks

3.1.30 EdsCreateMemoryStream

Description

Creates a stream in the memory of a host computer.
In the case of writing in excess of the allocated buffer size, the memory is automatically extended.

Syntax

```
EDSError    EdsCreateMemoryStream ( EdsUInt32 inBufferSize,
                                    EdsStreamRef* outStreamRef     )
```

Parameters

inBufferSize

Designate the buffer size to allocate. Because the size will be extended automatically as needed, designate 0 if the buffer size is unknown.

outStreamRef

On normal completion, a pointer is specified to the stream object that was created.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

EdsCreateFileStream, EdsWrite, EdsRead, and EdsRelease

Note

• The stream you create must be released after use by means of EdsRelease.

3.1.31 EdsCreateMemoryStreamFromPointer

Description

Creates a stream from the memory buffer you prepare. Unlike the buffer size of streams created by means of EdsCreateMemoryStream, the buffer size you prepare for streams created this way does not expand.

Syntax

```
EdsError    EDSAPI    EdsCreateMemoryStreamFromPointer (
                                  EdsVoid          *inUserBuffer,
                                  EdsUInt32        inBufferSize,
                                  EdsStreamRef     *outStream
);
```

Parameters

inUserBuffer

Pointer to the buffer you have prepared. Streams created by means of this API lead to this buffer.

inBufferSize

Designate the buffer size.

outStream

On normal completion, returns the stream to the designated buffer. Designate the reference to the EdsStreamRef type variable (that is, the address) as an argument.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

Revision History/Date		Corrections	Reviser	Remarks

• Related APIs
EdsCreateMemoryStream, EdsCreateFileStream, EdsCreateFileStreamEx, EdsWrite, and EdsRelease

Note

• The size of streams created by means of this API does not change. Be careful to ensure that access to the created stream does not exceed the available space.

3.1.32 EdsGetPointer

Description

Gets the pointer to the start address of memory managed by the memory stream.
As the EDSDK automatically resizes the buffer, the memory stream provides you with the same access methods as for the file stream. If access is attempted that is excessive with regard to the buffer size for the stream, data before the required buffer size is allocated is copied internally, and new writing occurs. Thus, the buffer pointer might be switched on an unknown timing. Caution in use is therefore advised.

Syntax

```
EdsError EDSAPI EdsGetPointer(
                        EdsStreamRef    inStream,
                        EdsVoid         **outPointer
) ;
```

Parameters

inStream

Designate the memory stream for the pointer to retrieve.

outPointer

If successful, returns the pointer to the buffer written in the memory stream.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs
EdsCreateMemoryStream, EdsCreateFileStream, EdsCreateFileStreamEx, EdsWrite, and EdsRelease

Note

• The buffer pointer may be switched on an unknown timing. Thus, some risk is posed by using this API so that saved pointers are saved and used in alternation. Caution in use is therefore advised.

3.1.33 EdsRead

Description

Reads data the size of inReadSize into the outBuffer buffer, starting at the current read or write position of the stream. The size of data actually read can be designated in outReadSize.

Syntax

```
EdsError    EdsRead(
                    EdsStreamRef inStreamRef,
                    EdsUInt32           inReadSize,
```

Revision History/Date		Corrections	Reviser	Remarks

EdsVoid *outBuffer,
EdsUInt32 *outReadSize)

Parameters

inStreamRef
Designate the file or memory stream.

inReadSize
Designate the size of data to read.

outBuffer
On normal completion, specifies the buffer storing read data.

outReadSize
Specifies a pointer to the variable for receiving the size of data actually read.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs
EdsCreateMemoryStream, EdsCreateFileStream, EdsCreateFileStreamEx, EdsWrite, and EdsRelease

Note

• If reading is successful, the read or write position in the stream is moved ahead an amount corresponding to the size of data read.

3.1.34 EdsWrite

Description

Writes data of a designated buffer to the current read or write position of the stream.

Syntax

EdsError EdsWrite(EdsStreamRef inStreamRef, EdsUInt32 inWriteSize,
Const EdsVoid* inBuffer, EdsUInt32 *outWrittenSize)

Parameters

inStreamRef
Designate the destination stream for writing. The stream object must be retrieved in advance.

inWriteSize
Designate the size of data to write from the buffer.

inBuffer
Designate a pointer to the data to write.

outWrittenSize
Specifies a pointer to the variable for receiving the size of data actually written.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs
EdsCreateMemoryStream, EdsCreateFileStream, EdsCreateFileStreamEx, EdsRead, and EdsRelease

Note

Revision History/Date		Corrections	Reviser	Remarks

• If writing is successful, the read or write position in the stream is moved ahead an amount corresponding to the size of data written.

3.1.35 EdsSeek

Description

Moves the read or write position of the stream (that is, the file position indicator).

Syntax

EdsError EdsSeek(EdsStreamRef inStreamRef, EdsInt32 inSeekOffset, EdsSeekOrigin inSeekOrigin)

Parameters

inStreamRef

Designate the stream object for this operation.

inSeekOffset

Designate the number of bytes to move the file position indicator.

inSeekOrigin

Designate the origin for moving from the read or write position. Designate any of the following, as defined in enum EdsSeekOrigin.

Enum EdsSeekOrigin <defined location>EDSDKTypes.h

InSeekOrigin	Description
kEdsSeek_Begin	Moves the file position indicator from the beginning of the stream forward by inOffset bytes.
kEdsSeek_Cur	Moves the file position indicator from the current position in the stream forward by inOffset bytes.
kEdsSeek_End	Moves the file position indicator from the end of the stream by inOffset bytes. To move toward the beginning, designate a negative value. Positive values will move the indicator beyond the end of the file.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs

EdsCreateMemoryStream, EdsCreateFileStream, EdsCreateFileStreamEx, EdsRead, and EdsWrite

3.1.36 EdsGetPosition

Description

Gets the current read or write position of the stream (that is, the file position indicator).

Syntax

EdsError EdsGetPosition(EdsStreamRef inStreamRef, EdsUInt32* outPosition)

Parameters

inStreamRef

Designate the destination stream for getting the position.

Revision History/Date		Corrections	Reviser	Remarks

outPosition
On normal completion, specifies a pointer to the variable for receiving the current read or write position of the stream (that is, to the offset position from the beginning of the stream). (The beginning of the stream is 0.)

Return Values
Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also
• Related APIs
EdsCreateMemoryStream, EdsCreateFileStream, EdsCreateFileStreamEx, EdsRead, EdsWrite, and EdsSeek

Note
• The stream's initial read or write position is 0. If EdsWrite or EdsRead is used to write or read from the stream, the indicator is moved an amount corresponding to that size in the positive direction.
• When intentionally changing the read or write position of the stream, use EdsSeek.

3.1.37 EdsGetLength
Description
Gets the stream size.

Syntax
EdsError EdsGetLength(EdsStreamRef inStreamRef, EdsUInt32 *outLength)

Parameters
inStreamRef
Designate the stream object for this operation.
outLength
Specifies the pointer to the variable for receiving the number of bytes of the stream.

Return Values
Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also
• Related APIs
EdsCreateMemoryStream, EdsCreateFileStream, and EdsCreateFileStreamEx

3.1.38 EdsCopyData
Description
Copies data from the copy source stream to the copy destination stream.
The read or write position of the data to copy is determined from the current file read or write position of the respective stream.
After this API is executed, the read or write positions of the copy source and copy destination streams are moved an amount corresponding to inWriteSize in the positive direction.

Syntax
EdsError EdsCopyData(
EdsStreamRef inStreamRef, EdsUInt32 inWriteSize,

Revision History/Date		Corrections	Reviser	Remarks

EdsStreamRef outStreamRef)

Parameters

inStreamRef

Designate the source stream for copying.

inWriteSize

Designate the number of bytes to copy .

outStreamRef

Designate the destination stream for copying.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs

EdsCreateMemoryStream, EdsCreateFileStream, EdsCreateFileStreamEx, EdsRead, EdsWrite, EdsSeek, and EdsGetPosition

3.1.39 EdsCreateImageRef

Description

Creates an image object from an image file.
Without modification, stream objects cannot be worked with as images. Thus, when extracting images from image files, you must use this API to create image objects.
The image object created this way can be used to get image information (such as the height and width, number of color components, and resolution), thumbnail image data, and the image data itself.

Syntax

```
EdsError  EdsCreateImageRef(  EdsStreamRef  inStreamRef,
                              EdsImageRef  *outImageRef  )
```

Parameters

inStreamRef

Designate the image file (or image data in the memory stream).

outImageRef

Specifies the pointer to the variable for receiving the image object.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs

EdsCreateStream, EdsGetImageInfo, and EdsGetImage, EdsRelease

3.1.40 EdsGetImageInfo

Description

Gets image information from a designated image object.
Here, image information means the image width and height, number of color components, resolution, and effective image area.

Revision History/Date		Corrections	Reviser	Remarks

Syntax

EdsError EdsGetImageInfo(
EdsImageRef inImageRef, EdsImageSource inImageSource,
EdsImageInfo* outImageInfo)

Parameters

inStreamRef

Designate the object for which to get image information.

inImageSource

Of the various image data items in the image file, designate the type of image data representing the information you want to get. Designate the image as defined in Enum EdsImageSource.

Enum EdsImageSource <defined location>EDSDKTypes.h

Value	Description
kEdsImageSrc_FullView	The image itself (a full-sized image)
kEdsImageSrc_Thumbnail	A thumbnail image
kEdsImageSrc_Preview	A preview image
kEdsImageSrc_RAWThumbnail	A RAW thumbnail image
kEdsImageSrc_RAWFullView	A RAW full-sized image

outImageInfo

Stores the image data information designated in inImageSource.

EdsImageInfo constituent elements	Type	Description
width	EdsUInt32	Width (in pixels)
height	EdsUInt32	Height (in pixels)
numOfComponents	EdsUInt32	Number of color components
componentDepth	EdsUInt32	Resolution (8-bit or 16-bit) Note: Image files may contain image data of mixed resolutions.
effectiveRect	EdsRect	Effective image area (This means the area excluding the black bands on the top and bottom of the thumbnail image.)
Reserved	EdsUInt32	Reserved

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

- Related APIs
  EdsCreateImageRef and EdsGetImage
- For information on data types of the EDSDK, see "Data Types Used by the APIs" in the Appendix.

3.1.41 EdsGetImage

Description

Gets designated image data from an image file, in the form of a designated rectangle.
Returns uncompressed results for JPEG compressed images and processed results in the designated pixel order (RGB, Top-down BGR, and so on) for RAW images. Additionally, by designating the input/output rectangle, it is possible to get reduced, enlarged, or partial images. However, because images corresponding to the designated output rectangle are always returned by the SDK, the SDK

Revision History/Date		Corrections	Reviser	Remarks

does not take the aspect ratio into account. To maintain the aspect ratio, you must keep the aspect ratio in mind when designating the rectangle.

Syntax

```
EdsError  EDSAPI  EdsGetImage(
            EdsImageRef          inImageRef,
            EdsImageSource       inImageSource,
            EdsTargetImageType   inImageType,
            EdsRect              inSrcRect,
            EdsSize              inDstSize,
            EdsStreamRef         outStreamRef
) ;
```

Parameters

inImageRef

Designate the image object for which to get the image data.

inImageSource

Designate the type of image data to get from the image file (thumbnail, preview, and so on). Designate values as defined in Enum EdsImageSource.

Enum EdsImageSource <defined location>EDSDKTypes.h

Value	Description
kEdsImageSrc_FullView	The image itself (a full-sized image)
kEdsImageSrc_Thumbnail	A thumbnail image
kEdsImageSrc_Preview	A preview image (displayed on the back screen of the camera)
kEdsImageSrc_RAWThumbnail	A RAW thumbnail image
kEdsImageSrc_RAWFullView	A RAW full-sized image

inImageType

Designate the output image type. Because the output format of EdGetImage may only be RGB, only **kEdsTargetImageType_RGB or kEdsTargetImageType_RGB16** can be designated. However, image types exceeding the resolution of inImageSource cannot be designated.

Example: Suppose the source image resolution (componentDepth) retrieved by means of **EdsGetImageInfo()** is 8 bits

→ The resolution that can be retrieved by means of EdsGetImage () is also 8 bits

→ Thus, only **kEdsTargetImageType_RGB** is available.

EdsTargetImageType <defined location>EDSDKTypes.h

Value	Description
kEdsTargetImageType _RGB	8-bit RGB, chunky format
kEdsTargetImageType _RGB16	16-bit RGB, chunky format

inSrcRect

Designate the coordinates and size of the rectangle to be retrieved (processed) from the source image.

inDstSize

Designate the rectangle size for output.

outStreamRef

Designate the memory or file stream for output of the image.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

Revision History/Date		Corrections	Reviser	Remarks

See Also

- Related APIs
  EdsCreateImageRef and EdsGetImageInfo

Note

- To maintain the aspect ratio, you must keep the aspect ratio in mind when designating a rectangle.
- In calculating the data size of the output file, the original image data resolution is not used. Instead, the resolution of the image type designated by inImageType is used. For example, the calculation for kEdsTargetImageType_RGB is 3 (R, G, and B) x 8 (resolution) x width x height ÷ 8 (bytes). Similarly, kEdsTargetImageType_RGB16 is calculated by 3 x 16 x width x height ÷ 8 (bytes).

3.1.42 EdsSaveImage

Description

Saves as a designated image type after RAW processing.
When saving with JPEG compression, the JPEG quality setting applies with respect to EdsOptionRef.

Syntax

```
EdsError EDSAPI EdsSaveImage(
                    EdsImageRef             inImageRef,
                    EdsTargetImageType      inImageType,
                    EdsSaveImageSetting     inSaveSetting,
                    EdsStreamRef            outStreamRef
) ;
```

Parameters

inImageRef

Designate the image object for which to produce the file.

inImageType

Designate the image type to produce. Designate the following image types.

Enum EdsTargetImageType <defined location>EDSDKTypes.h

Value	Description
kEdsTargetImageType _Jpeg	JPEG
kEdsTargetImageType _TIFF	8-bit TIFF
kEdsTargetImageType _TIFF16	16-bit TIFF

inSaveSetting

Designate saving options, such as JPEG quality.

EdsSaveImageSetting <defined location>EDSDKTypes.h

EdsSaveImageSetting constituent elements	Type	Description
JPEGQuality	EdsUInt32	Image quality for JPEG compression 1 (rough) to 10 (fine)
iccProfileStream	EdsStreamRef	ICC profile stream
reserved	EdsUInt32	Reserved

outStreamRef

Specifies the output file stream. The memory stream cannot be specified here.

Revision History/Date		Corrections	Reviser	Remarks

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

- Related APIs
  EdsCreateFileStream, EdsCreateFileStreamEx, EdsRead, and EdsWrite
- For information on data types of the EDSDK, see "Data Types Used by the APIs" in the Appendix.

3.1.43 EdsCacheImage

Description

Switches a setting on and off for creation of an image cache in the SDK for a designated image object during extraction (processing) of the image data. Creating the cache increases the processing speed, starting from the second time.

Syntax

```
EdsError  EDSAPI  EdsCacheImage(
                        EdsImageRef     inImageRef,
                        EdsBool         inUseCache
) ;
```

Parameters

inImageRef

Designate the image object.

inUseCache

TRUE: Image cache ON
FALSE: Image cache OFF

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

- Related APIs
  EdsGetImage and EdsSaveImage

Note

- If the image cache is on, a corresponding amount of resources are consumed. If fast processing is not required, use the EDSDK with the cache off.

3.1.44 EdsReflectImageProperty

Description

Incorporates image object property changes (effected by means of EdsSetPropertyData) in the stream.

Syntax

```
EdsError  EDSAPI  EdsReflectImageProperty(
                        EdsImageRef     inImageRef
) ;
```

Parameters

inImageRef

Revision History/Date		Corrections	Reviser	Remarks

Designate the image object.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs
EdsSetPropertyData

3.1.45 EdsSetCameraAddedHandler

Description

Registers a callback function for when a camera is detected.

Syntax

```
EdsError    EdsSetCameraAddedHandler (
                    EdsCameraAddedHandler inCameraAddedHandler,
                    EdsVoid*    inContext    )
```

Parameters

inCameraAddedHandler

Designate the pointer to the callback function called when a camera is detected.
You must implement the callback function registered this way following a prescribed type definition.
The callback function type is defined as follows.

Syntax

```
typedef    EdsError
   ( EDSCALLBACK * EdsCameraAddedHandler)(EdsVoid *inContext )
```

Parameters

inContext

Passes data for the application designated by **EdsSetCameraAddedHandler**.

Return Values

Returns EDS_ERR_OK if successful. Otherwise, ensure the implementation returns an appropriate error code. (See the EDS Error Lists).

inContext

Designate application information to be passed by means of the callback function. Any data needed for your application can be passed.
In multithreaded environments, the callback function is executed by a thread exclusively for the event. Use it appropriately, as in designating the `this` pointer to pass data to UI threads.
Designate a NULL pointer if it is not needed.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs
EdsSetPropertyEventHandler, EdsSetObjectEventHandler, EdsSetCameraStateEventHandler, and EdsSetProgressCallback

Revision History/Date		Corrections	Reviser	Remarks

3.1.46 EdsSetObjectEventHandler

Description

Registers a callback function for receiving status change notification events for objects on a remote camera. Here, object means volumes representing memory cards, files and directories, and shot images stored in memory, in particular.

Syntax

```
EdsError  EdsSetObjectEventHandler(  EdsCameraRef  inCameraRef,
                                     EdsObjectEvent  inEvent,
                                     EdsObjectEventHandler  inObjectEventHandler,
                                     EdsVoid  *inContext  )
```

Parameters

inCameraRef

Designate the camera object.

inEvent

Designate one or all events to be supplemented. To designate all events, use kEdsObjectEvent_All. For details on events that can be designated, refer to the section on object-related events in the event lists of Asynchronous Events.

inObjectEventHandler

Designate the pointer to the callback function for receiving object-related camera events. The callback function registered here is called by the EDSDK when the event is received.
To cancel supplementation of the event designated in the event type, designate NULL in this argument.
You must implement the callback function registered this way following a prescribed type definition. The callback function type for object-related events is defined as follows.

Syntax

```
typedef EdsError  ( EDSCALLBACK *EdsObjectEventHandler)(
                                  EdsObjectEvent  inEvent,
                                  EdsBaseRef  inRef,

                                  EdsVoid  *inContext) ;
```

Parameters

inEvent

Indicate the event type supplemented. Designate one of the event types for supplementation, as designated by EdsSetObjectEventHandler. Events that occur can be determined based on the event type.

inRef

Returns a reference to objects created by the event.

inContext

Passes inContext without modification, as designated as an **EdsSetObjectEventHandler** argument.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

inContext

Designate application information to be passed by means of the callback function. Any data needed for your application can be passed.

Revision History/Date		Corrections	Reviser	Remarks

In multithreaded environments, the callback function is executed by a thread exclusively for the event. Use it appropriately, as in designating the `this` pointer to pass data to UI threads. Designate a NULL pointer if it is not needed.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

- Related APIs
  EdsSetCameraAddedHandler, EdsSetPropertyEventHandler, EdsSetCameraStateEventHandler, and EdsSetProgressCallback
- For details on asynchronous events, refer to "Overview" and "Asynchronous Events."

Note

- To release the event handler for events of the designated type, designate NULL in the argument of inObjectEventHandler. (The event will not occur.)

Example

- See Sample 1.

3.1.47 EdsSetPropertyEventHandler

Description

Registers a callback function for receiving status change notification events for property states on a camera.

Syntax

```
EdsError EDSAPI EdsSetPropertyEventHandler(
                EdsCameraRef            inCameraRef,
                EdsPropertyEvent        inEvnet,
                EdsPropertyEventHandler inPropertyEventHandler,
                EdsVoid*                inContext );
```

Parameters

inCameraRef

Designate the camera object.

inEvent

Designate one or all events to be supplemented. To designate all events, use kEdsPropertyEvent_All. For details on events that can be designated, refer to the section on property-related events in the event lists of Asynchronous Events.

inPropertyEventHandler

Designate the pointer to the callback function for receiving property-related camera events. The callback function registered here is called by the EDSDK when the event is received.
To cancel supplementation of the event designated in the event type, designate NULL in this argument.
You must implement the callback function registered this way following a prescribed type definition. The callback function type for property-related events is defined as follows.

Syntax

```
typedef EdsError    ( EDSCALLBACK * EdsPropertyEventHandler )(
                        EdsPropertyEvent        inEvent,
                        EdsPropertyID           inPropertyID,
```

Revision History/Date		Corrections	Reviser	Remarks

EdsUInt32 inParam,
EdsVoid *inContext);

Parameters

inEvent
Indicate the event type supplemented. Designate one of the event types subject to supplementation, as designated by EdsSetPropertyEventHandler. Events that occur can be determined based on the event type.

inPropertyID
Returns the property ID created by the event.

inParam
Used to identify information created by the event for custom function (CF) properties or other properties that have multiple items of information.

inContext
Passes inContext without modification, as designated as an **EdsSetPropertyEventHandler** argument.

Return Values
Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

inContext
Designate application information to be passed by means of the callback function. Any data needed for your application can be passed.
In multithreaded environments, the callback function is executed by a thread exclusively for the event. Use it appropriately, as in designating the `this` pointer to pass data to UI threads. Designate a NULL pointer if it is not needed.

Return Values
Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also
- Related APIs
  EdsSetCameraAddedHandler, EdsObjectEventHandler, EdsSetCameraStateEventHandler, and EdsSetProgressCallback
- For details on asynchronous events, refer to "Overview" and "Asynchronous Events."

Note
- To release the event handler for events of the designated type, designate NULL in the argument of inPropertyEventHandler. (The event will not occur.)

Example
- See Sample 1.

3.1.48 EdsSetCameraStateEventHandler

Description
Registers a callback function for receiving status change notification events for camera objects.

Syntax
EdsError EDSAPI EdsSetCameraStateEventHandler(
EdsCameraRef inCameraRef,
EdsStateEvent inEvnet,

Revision History/Date		Corrections	Reviser	Remarks

```
                    EdsStateEventHandler    inStateEventHandler,
                    EdsVoid*                inContext );
```

Parameters

inCameraRef

Designate the camera object.

inEvent

Designate one or all events to be supplemented. To designate all events, use kEdsStateEvent_All. For details on events that can be designated, refer to the section on events related to camera states in the event lists of Asynchronous Events.

inStateEventHandler

Designate the pointer to the callback function for receiving events related to camera object states. The callback function registered here is called by the EDSDK when the event is received.
To cancel supplementation of the event designated in the event type, designate NULL in this argument.
You must implement the callback function registered this way following a prescribed type definition. The callback function type for events related to camera states is defined as follows.

Syntax

```
typedef EdsError    ( EDSCALLBACK *EdsStateEventHandler )(
                                    EdsStateEvent    inEvent,
                                    EdsUInt32        inEventData,
                                    EdsVoid          *inContext );
```

Parameters

inEvent

Indicate the event type supplemented. Designate one of the event types subject to supplementation, as designated by EdsSetPropertyEventHandler. Events that occur can be determined based on the event type.

inEventData

Pointer to the event data. The content designated here varies depending on the property type. For details, see Property Details.

inContext

Passes inContext without modification, as designated as an **EdsSetCameraStateEventHandler** argument.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

inContext

Designate application information to be passed by means of the callback function. Any data needed for your application can be passed.
In multithreaded environments, the callback function is executed by a thread exclusively for the event. Use it appropriately, as in designating the `this` pointer to pass data to UI threads. Designate a NULL pointer if it is not needed.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs
EdsSetCameraAddedHandler, EdsObjectEventHandler, EdsSetObjectEventHandler, and

Revision History/Date		Corrections	Reviser	Remarks

EdsSetProgressCallback
• For details on asynchronous events, refer to "Overview" and "Asynchronous Events."

Note

• To release the event handler for events of the designated type, designate NULL in the argument of inStateEventHandler. (The event will not occur.)

3.1.49 EdsSetProgressCallback

Description

Register a progress callback function.
An event is received as notification of progress during processing that takes a relatively long time, such as downloading files from a remote camera. If you register the callback function, the EDSDK calls the callback function during execution or on completion of the following APIs. This timing can be used in updating on-screen progress bars, for example.

APIs for which the progress callback function is valid
EdsCopyData
EdsDownload
EdsGetImage
EdsSaveImage

Syntax

```
EdsError EdsSetProgressCallback(
                    EdsBaseRef          inRef,
                    EdsProgressFunc     inProgressCallback,
                    EdsProgressOption   inProgressOption,
                    EdsVoid*            inContext)
```

Parameters

inRef

Designate the relevant object.
EdsImageRef or EdsStreamRef are the objects of APIs for which progress callback registration is valid.

inProgressCallback

Designate a pointer to the progress callback function.
The progress callback function type is defined as follows.

Syntax

```
typedef  EdsError( EDSCALLBACK * EdsProgressCallback )(
                    EdsUInt32       inPercent,
                    EdsVoid         *inContext,
                    EdsBool         *outCancel )
```

Parameters

inPercent

Indicates the progress in a range of 0 –100%. Value range: 0 to 100

inContext

The application information designated by EdsSetProgressCallback.

outCancel

To cancel processing in progress, set this variable to TRUE.
For example, if this argument is set to TRUE during file transfer from the camera, the

Revision History/Date		Corrections	Reviser	Remarks

EDSDK notifies the camera that file transfer has been canceled, and transfer of those files is canceled.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

inProgressOption

Options when this callback function is called are defined in Enum EdsProgressOption.

Enum EdsProgressOption <defined location>EDSDKTypes.h

Value	Description
kEdsProgressOption_NoReport	Do not call a progress callback function.
kEdsProgressOption_Done	Call a progress callback function when the progress reaches 100%.
kEdsProgressOption_Periodically	Call a progress callback function periodically.

inContext

Application information, passed in the argument when the callback function is called. Any information required for your program may be added.

Return Values

Returns EDS_ERR_OK if successful. In other cases, see the EDS Error Lists.

See Also

• Related APIs
EdsSetCameraAddedHandler and EdsSetObjectEventHandler

Note

• To release the event handler for events of the designated type, designate NULL in the argument of inStateEventHandler. (The event will not occur.)

Revision History/Date		Corrections	Reviser	Remarks

3.2 EDS Error Lists

As return values, EDSDK APIs return error codes defined as follows.

For each API, the return values mainly used are identified based on API characteristics. However, the principal factors that actually caused the problems are specified as error codes. Thus, all error codes may be specified in return values.

3.2.1 General errors

Error Type	*Notes*
EDS_ERR_UNIMPLEMENTED	Not implemented
EDS_ERR_INTERNAL_ERROR	Internal error
EDS_ERR_MEM_ALLOC_FAILED	Memory allocation error
EDS_ERR_MEM_FREE_FAILED	Memory release error
EDS_ERR_OPERATION_CANCELLED	Operation canceled
EDS_ERR_INCOMPATIBLE_VERSION	Version error
EDS_ERR_NOT_SUPPORTED	Not supported
EDS_ERR_UNEXPECTED_EXCEPTION	Unexpected exception
EDS_ERR_PROTECTION_VIOLATION	Protection violation
EDS_ERR_MISSING_SUBCOMPONENT	Missing subcomponent
EDS_ERR_SELECTION_UNAVAILABLE	Selection unavailable

3.2.2 File access errors

Error Type	*Notes*
EDS_ERR_FILE_IO_ERROR	IO error
EDS_ERR_FILE_TOO_MANY_OPEN	Too many files open
EDS_ERR_FILE_NOT_FOUND	File does not exist
EDS_ERR_FILE_OPEN_ERROR	Open error
EDS_ERR_FILE_CLOSE_ERROR	Close error
EDS_ERR_FILE_SEEK_ERROR	Seek error
EDS_ERR_FILE_TELL_ERROR	Tell error
EDS_ERR_FILE_READ_ERROR	Read error
EDS_ERR_FILE_WRITE_ERROR	Write error
EDS_ERR_FILE_PERMISSION_ERROR	Permission error
EDS_ERR_FILE_DISK_FULL_ERROR	Disk full
EDS_ERR_FILE_ALREADY_EXISTS	File already exists
EDS_ERR_FILE_FORMAT_UNRECOGNIZED	Format error
EDS_ERR_FILE_DATA_CORRUPT	Invalid data
EDS_ERR_FILE_NAMING_NA	File naming error

3.2.3 Directory errors

Error Type	*Notes*
EDS_ERR_DIR_NOT_FOUND	Directory does not exist
EDS_ERR_DIR_IO_ERROR	I/O error
EDS_ERR_DIR_ENTRY_NOT_FOUND	No file in directory
EDS_ERR_DIR_ENTRY_EXISTS	File in directory
EDS_ERR_DIR_NOT_EMPTY	Directory full

Revision History/Date		Corrections	Reviser	Remarks

3.2.4 Property errors

Error Type	*Notes*
EDS_ERR_PROPERTIES_UNAVAILABLE	Property (and additional property information) unavailable
EDS_ERR_PROPERTIES_MISMATCH	Property mismatch
EDS_ERR_PROPERTIES_NOT_LOADED	Property not loaded

3.2.5 Function parameter errors

Error Type	*Notes*
EDS_ERR_INVALID_PARAMETER	Invalid function parameter
EDS_ERR_INVALID_HANDLE	Handle error
EDS_ERR_INVALID_POINTER	Pointer error
EDS_ERR_INVALID_INDEX	Index error
EDS_ERR_INVALID_LENGTH	Length error
EDS_ERR_INVALID_FN_POINTER	FN pointer error
EDS_ERR_INVALID_SORT_FN	Sort FN error

3.2.6 Device errors

Error Type	*Notes*
EDS_ERR_DEVICE_NOT_FOUND	Device not found
EDS_ERR_DEVICE_BUSY	Device busy Note: If a device busy error occurs, reissue the command after a while. The camera will become unstable.
EDS_ERR_DEVICE_INVALID	Device error
EDS_ERR_DEVICE_EMERGENCY	Device emergency
EDS_ERR_DEVICE_MEMORY_FULL	Device memory full
EDS_ERR_DEVICE_INTERNAL_ERROR	Internal device error
EDS_ERR_DEVICE_INVALID_PARAMETER	Device parameter invalid
EDS_ERR_DEVICE_NO_DISK	No disk
EDS_ERR_DEVICE_DISK_ERROR	Disk error
EDS_ERR_DEVICE_CF_GATE_CHANGED	The CF gate has been changed
EDS_ERR_DEVICE_DIAL_CHANGED	The dial has been changed
EDS_ERR_DEVICE_NOT_INSTALLED	Device not installed
EDS_ERR_DEVICE_STAY_AWAKE	Device connected in awake mode
EDS_ERR_DEVICE_NOT_RELEASED	Device not released

3.2.7 Stream errors

Error Type	*Notes*
EDS_ERR_STREAM_IO_ERROR	Stream I/O error
EDS_ERR_STREAM_NOT_OPEN	Stream open error
EDS_ERR_STREAM_ALREADY_OPEN	Stream already open
EDS_ERR_STREAM_OPEN_ERROR	Failed to open stream
EDS_ERR_STREAM_CLOSE_ERROR	Failed to close stream
EDS_ERR_STREAM_SEEK_ERROR	Stream seek error
EDS_ERR_STREAM_TELL_ERROR	Stream tell error

Revision History/Date		Corrections	Reviser	Remarks

EDS_ERR_STREAM_READ_ERROR	Failed to read stream
EDS_ERR_STREAM_WRITE_ERROR	Failed to write stream
EDS_ERR_STREAM_PERMISSION_ERROR	Permission error
EDS_ERR_STREAM_COULDNT_BEGIN_THREAD	Could not start reading thumbnail
EDS_ERR_STREAM_BAD_OPTIONS	Invalid stream option
EDS_ERR_STREAM_END_OF_STREAM	Invalid stream termination

3.2.8 Communication errors

Error Type	*Notes*
EDS_ERR_COMM_PORT_IS_IN_USE	Port in use
EDS_ERR_COMM_DISCONNECTED	Port disconnected
EDS_ERR_COMM_DEVICE_INCOMPATIBLE	Incompatible device
EDS_ERR_COMM_BUFFER_FULL	Buffer full
EDS_ERR_COMM_USB_BUS_ERR	USB bus error

3.2.9 Camera UI lock/unlock errors

Error Type	*Notes*
EDS_ERR_USB_DEVICE_LOCK_ERROR	Failed to lock the UI
EDS_ERR_USB_DEVICE_UNLOCK_ERROR	Failed to unlock the UI

3.2.10 STI/WIA errors

Error Type	*Notes*
EDS_ERR_STI_UNKNOWN_ERROR	Unknown STI
EDS_ERR_STI_INTERNAL_ERROR	Internal STI error
EDS_ERR_STI_DEVICE_CREATE_ERROR	Device creation error
EDS_ERR_STI_DEVICE_RELEASE_ERROR	Device release error
EDS_ERR_DEVICE_NOT_LAUNCHED	Device startup failed

3.2.11 Other general error

Error Type	*Notes*
EDS_ERR_ENUM_NA	Enumeration terminated (there was no suitable enumeration item)
EDS_ERR_INVALID_FN_CALL	Called in a mode when the function could not be used
EDS_ERR_HANDLE_NOT_FOUND	Handle not found
EDS_ERR_INVALID_ID	Invalid ID
EDS_ERR_WAIT_TIMEOUT_ERROR	Timeout
EDS_ERR_LAST_GENERIC_ERROR_PLUS_ONE	Not used.

3.2.12 PTP errors

Error Type	*Notes*
EDS_ERR_SESSION_NOT_OPEN	Session open error
EDS_ERR_INVALID_TRANSACTIONID	Invalid transaction ID

Revision History/Date		Corrections	Reviser	Remarks

EDS_ERR_INCOMPLETE_TRANSFER	Transfer problem
EDS_ERR_INVALID_STRAGEID	Storage error
EDS_ERR_DEVICEPROP_NOT_SUPPORTED	Unsupported device property
EDS_ERR_INVALID_OBJECTFORMATCODE	Invalid object format code
EDS_ERR_SELF_TEST_FAILED	Failed self-diagnosis
EDS_ERR_PARTIAL_DELETION	Failed in partial deletion
EDS_ERR_SPECIFICATION_BY_FORMAT_UNSUPPORTED	Unsupported format specification
EDS_ERR_NO_VALID_OBJECTINFO	Invalid object information
EDS_ERR_INVALID_CODE_FORMAT	Invalid code format
EDS_ERR_UNKNOWN_VENDER_CODE	Unknown vendor code
EDS_ERR_CAPTURE_ALREADY_TERMINATED	Capture already terminated
EDS_ERR_INVALID_PARENTOBJECT	Invalid parent object
EDS_ERR_INVALID_DEVICEPROP_FORMAT	Invalid property format
EDS_ERR_INVALID_DEVICEPROP_VALUE	Invalid property value
EDS_ERR_SESSION_ALREADY_OPEN	Session already open
EDS_ERR_TRANSACTION_CANCELLED	Transaction canceled
EDS_ERR_SPECIFICATION_OF_DESTINATION_UNSUPPORTED	Unsupported destination specification
EDS_ERR_UNKNOWN_COMMAND	Unknown command
EDS_ERR_OPERATION_REFUSED	Operation refused
EDS_ERR_LENS_COVER_CLOSE	Lens cover closed

Revision History/Date		Corrections	Reviser	Remarks

4. Asynchronous Events

In the case of asynchronous events, notify the host computer of changes, such as changes in the state of properties of remote cameras.

To enable an application to receive issued events, you must prepare callback functions for event reception and register them in the EDSDK by means of EdsSetPropertyEventHandler, EdsSetObjectEventHandler, EdsSetCameraStateEventHandler, EdsSetCameraAddedHandler, EdsSetProgressCallback, or other APIs for configuring callback functions.

For details on callback function types, see the parameters information of the APIs for callback function configuration.

This section describes events that can be retrieved by callback functions registered using EdsSetPropertyEventHandler, EdsSetObjectEventHandler, and EdsSetCameraStateEventHandler in particular.

4.1 Event Lists

4.1.1 Object-related events

Events
Notification of file creation
Notification of file deletion
Notification of changes in file information
Notification of changes in the volume information of recording media
Notification of requests to update volume information
Notification of requests to update folder information
Notification of file transfer requests
Notification of direct transfer requests
Notification of requests to cancel direct transfer

4.1.2 Property-related events

Events
Notification of property state changes
Notification of state changes in configurable property values

4.1.3 State-related events

Events
Notification of camera disconnection
Notification of changes in job states
Notification of warnings when the camera will shut off
Notification that the camera will remain on for a longer period
Notification of remote release failure
Notification of internal SDK errors

Revision History/Date		Corrections	Reviser	Remarks

4.2 Event Details

Events are explained in the following format.

4.2.xx EventID

Event ID of the issued event. Used to distinguish event types in callback functions.

Description

Explains the event and cites related considerations.

Event Data

Event data passed as event callback function arguments.

Event Data	Data Type	Argument Name in the Callback Function
The nature of the data that is passed	The data type	The value passed as an argument

4.2.1 kEdsStateEvent_Shutdown (Notification of camera disconnection)

Description

Indicates that a camera is no longer connected to a computer, whether it was disconnected by unplugging a cord, opening the compact flash compartment, turning the camera off, auto shut-off, or by other means.

Event Data

Event Data	Data Type	Value of inParameter
None	–	–

4.2.2 kEdsPropertyEvent_PropertyChanged (Notification of property state changes)

Description

Notifies that a camera property value has been changed.
The changed property can be retrieved from event data.
The changed value can be retrieved by means of EdsGetPropertyData.

In the case of type 1 protocol standard cameras, notification of changed properties can only be issued for custom functions (CFn).
If the property type is 0x0000FFFF, the changed property cannot be identified. Thus, retrieve all required properties repeatedly.

Event Data

Event Data	Data Type	Value of inPropertyID
The property type	EdsPropertyID	A property ID

See Also

• For details on property IDs, see the Property Lists.

Revision History/Date		Corrections	Reviser	Remarks

4.2.3 kEdsPropertyEvent_PropertyDescChanged (Notification of state changes in configurable property values)

Description

Notifies of changes in the list of camera properties with configurable values.
The list of configurable values for property IDs indicated in event data can be retrieved by means of EdsGetPropertyDesc.

For type 1 protocol standard cameras, the property ID is identified as "Unknown" during notification. Thus, you must retrieve a list of configurable values for all properties and retrieve the property values repeatedly.
(For details on properties for which you can retrieve a list of configurable properties, see the description of EdsGetPropertyDesc).

Event Data

Event Data	Data Type	Value of inPropertyID
Property type for which the list of configurable values has changed	EdsPropertyID	Of the capture-related properties, those properties that have configurable values that can be retrieved; otherwise, "Unknown" (0x0000FFFF)

See Also

For details on property IDs, see the Property Lists.

4.2.4 kEdsObjectEvent_DirItemCreated (Notification of file creation)

Description

Notifies of the creation of objects such as new folders or files on a camera compact flash card or the like.
This event is generated if the camera has been set to store captured images simultaneously on the camera and a computer, for example, but not if the camera is set to store images on the computer alone.
Newly created objects are indicated by event data.

Because objects are not indicated for type 1 protocol standard cameras, (that is, objects are indicated as NULL), you must again retrieve child objects under the camera object to identify the new objects.

Event Data

Event Data	Data Type	Value of inRef
New directory or file object	EdsDirectoryItemRef	Pointer to the directory or file object

4.2.5 kEdsObjectEvent_DirItemRemoved (Notification of file deletion)

Description

Notifies of the deletion of objects such as folders or files on a camera compact flash card or the like.
Deleted objects are indicated in event data.

Because objects are not indicated for type 1 protocol standard cameras, you must again retrieve child objects under the camera object to identify deleted objects.

Event Data

Revision History/Date		Corrections	Reviser	Remarks

Event Data	Data Type	Value of inRef
Deleted directory or file object	EdsDirectoryItemRef	Pointer to the directory or file object

4.2.6 kEdsObjectEvent_DirItemInfoChanged (Notification of changes in file information)

Description

Notifies that information of DirItem objects has been changed.
Changed objects are indicated by event data.
The changed value can be retrieved by means of EdsGetDirectoryItemInfo.
Notification of this event is not issued for type 1 protocol standard cameras.

Event Data

Event Data	Data Type	Value of inRef
Changed directory or file object	EdsDirectoryItemRef	Pointer to the directory or file object

4.2.7 kEdsObjectEvent_DirItemContentChanged

Description

Notifies that header information has been updated, as for rotation information of image files on the camera.
If this event is received, get the file header information again, as needed.
This function is for type 2 protocol standard cameras only.

Event Data

Event Data	Data Type	Value of inRef
Changed file	EdsDirectoryItemRef	Pointer to the directory item object

Note

To retrieve image properties, you must obtain them from image objects after using DownloadImage or DownloadThumbnail.

4.2.8 kEdsObjectEvent_VolumeInfoChanged (Notification of changes in the volume information of recording media)

Description

Notifies that the volume object (memory card) state (VolumeInfo) has been changed.
Changed objects are indicated by event data.
The changed value can be retrieved by means of EdsGetVolumeInfo.

Notification of this event is not issued for type 1 protocol standard cameras.

Event Data

Event Data	Data Type	Value of inRef
Changed volume object	EdsVolumeRef	Pointer to the volume object

Revision History/Date		Corrections	Reviser	Remarks

4.2.9 kEdsObjectEvent_VolumeUpdateItems (Notification of requests to update volume information)

Description

Notifies if the designated volume on a camera has been formatted. If notification of this event is received, get sub-items of the designated volume again as needed.
Changed volume objects can be retrieved from event data.
Objects cannot be identified on cameras earlier than the D30 if files are added or deleted. Thus, these events are subject to notification.

Event Data

Event Data	Data Type	Value of inRef
Changed volume object	EdsVolumeRef	Pointer to the volume object

4.2.10 kEdsObjectEvent_FolderUpdateItems (Notification of requests to update folder information)

Description

Notifies if many images are deleted in a designated folder on a camera. If notification of this event is received, get sub-items of the designated folder again as needed.
Changed folders (specifically, directory item objects) can be retrieved from event data.

Event Data

Event Data	Data Type	Value of inRef
Changed folder	EdsDirectoryItemRef	Pointer to the directory item object

4.2.11 kEdsStateEvent_JobStatusChanged (Notification of changes in job states)

Description

Notifies of whether or not there are objects waiting to be transferred to a host computer.
This is useful when ensuring all shot images have been transferred when the application is closed.
Notification of this event is not issued for type 1 protocol standard cameras.

Event Data

Event Data	Data Type	Value of inParameter
Whether or not there are objects waiting to be transferred	EdsUInt32	1: There are objects to be transferred 0: There are no objects to be transferred

4.2.12 kEdsObjectEvent_DirItemRequestTransfer (Notification of file transfer requests)

Description

Notifies that there are objects on a camera to be transferred to a computer.
This event is generated after remote release from a computer or local release from a camera.
If this event is received, objects indicated in the event data must be downloaded. Furthermore, if the application does not require the objects, instead of downloading them, execute EdsDownloadCancel and release resources held by the camera.
The order of downloading from type 1 protocol standard cameras must be the order in which the events are received.

Event Data

Revision History/Date		Corrections	Reviser	Remarks

Event Data	Data Type	Value of inRef
Array of directories or file objects to be transferred	EdsDirectoryItemRef	Directory or file object

4.2.13 kEdsObjectEvent_DirItemRequestTransferDT (Notification of direct transfer requests)

Description

Notifies if the camera's direct transfer button is pressed.
If this event is received, objects indicated in the event data must be downloaded. Furthermore, if the application does not require the objects, instead of downloading them, execute EdsDownloadCancel and release resources held by the camera.
Notification of this event is not issued for type 1 protocol standard cameras.

Event Data

Event Data	Data Type	Value of inRef
Array of directories or file objects to be transferred directly	EdsDirectoryItemRef []	Array of directories and file objects

4.2.14 kEdsObjectEvent_DirItemCancelTransferDT (Notification of requests to cancel direct transfer)

Description

Notifies of requests from a camera to cancel object transfer if the button to cancel direct transfer is pressed on the camera.
If the parameter is 0, it means that cancellation of transfer is requested for objects still not downloaded, with these objects indicated by kEdsObjectEvent_DirItemRequestTransferDT.
Notification of this event is not issued for type 1 protocol standard cameras.

Event Data

Event Data	Data Type	Value of inRef
Array of directories or file objects for which to cancel transfer	EdsDirectoryItemRef []	Array of directories and file objects

4.2.15 kEdsStateEvent_WillSoonShutDown (Notification of warnings when the camera will shut off)

Description

Notifies that the camera will shut down after a specific period.
Generated only if auto shut-off is set.
Exactly when notification is issued (that is, the number of seconds until shutdown) varies depending on the camera model.
To continue operation without having the camera shut down, use EdsSendCommand to extend the auto shut-off timer. The time in seconds until the camera shuts down is returned as the initial value.

Event Data

Event Data	Data Type	Value of inParameter
Number of seconds until the camera shuts down	EdsUint32	Number of seconds

Revision History/Date		Corrections	Reviser	Remarks

4.2.16 kEdsStateEvent_ShutDownTimerUpdate (Notification that the camera will remain on for a longer period)

Description

As the counterpart event to kEdsStateEvent_WillSoonShutDown, this event notifies of updates to the number of seconds until a camera shuts down. After the update, the period until shutdown is model-dependent.

Event Data

Event Data	Data Type	Value of inParameter
None	–	–

4.2.17 kEdsStateEvent_CaptureError (Notification of remote release failure)

Description

Notifies that a requested release has failed, due to focus failure or similar factors.

Event Data

Event Data	Data Type	Value of inParameter
Error code	EdsUint32	Error code

Error codes received in the event data are as follows.

Error Code	Description
0x00000001	Shooting failure
0x00000002	The lens was closed
0x00000003	General errors from the shooting mode, such as errors from the bulb or mirror-up mechanism
0x00000004	Sensor cleaning
0x00000005	Error because the camera was set for silent operation (in PF21)
0x00000006	Prohibited settings using CFn-2, and no card inserted
0x00000007	Card error (including CARD-FULL/No.-FULL)
0x00000008	Write-protected

4.2.18 kEdsStateEvent_InternalError (Notification of internal SDK errors)

Description

Notifies of internal SDK errors.
If this error event is received, the issuing device will probably not be able to continue working properly, so cancel the remote connection.

Event Data

Event Data	Data Type	Value of inParameter
–	EdsUint32	Unspecified value

Revision History/Date		Corrections	Reviser	Remarks

5. Properties

Properties of camera and images objects can be retrieved and set by means of **EdsGetPropertyData**, **EdsSetPropertyData**, and other APIs.

For certain properties, if the target object is a camera, you can use the **EdsGetPropertyDesc** API to get the properties that can currently be set. For details, see the description of EdsGetPropertyDesc.

If the target object is an image, it has properties besides current settings values—speciifcally, properties that store settings values at the time the image was shot. Current property settings values are usually indicated, assuming you do not particularly need the previous values. However, by designating a property ID and an OR value for kEdsPropID_AtCapture_Flag in the arguments for **EdsGetPropertyData**, you can get the properties at the time of shooting. For details, see the description of kEdsPropID_AtCapture_Flag properties.

For the various properties there are, this section explains the objects they describe and what the properties mean.

5.1 Property Lists

Property IDs are listed below. <defined location>EDSDKTypes.h

■ Camera Setting Properties

Value	Description
0x00000002	Product name
0x00000003	Body ID
0x00000004	Owner
0x00000005	Manufacturer
0x00000006	For cameras, the system time; for images, the shooting time
0x00000007	Firmware version
0x00000008	Battery state: 0–100% or "AC"
0x00000009	Custom Function settings
0x0000000a	Personal Function settings
0x0000000b	Destination where image was saved
0x00000201	Set user-defined data in Set1, 2, or 3, or read it out
0x00000202	Set user-defined data in Set1, 2, or 3, or read it out
0x00000203	User picture style data

■ Image Properties

Value	Description
0x00000100	Stored image
0x00000101	Value representing compression when saved as a JPEG; 1 to 10 (cap)
0x00000102	Image orientation
0x00000103	ICC Profile data
0x00000104	Focus information
0x00000105	Digital exposure compensation
0x00000106	White balance (light source)
0x00000107	Color temperature setting value
0x00000108	White balance shift compensation
0x00000109	Contrast setting
0x0000010a	Saturation setting

Revision History/Date		Corrections	Reviser	Remarks

0x0000010b	Color tone setting
0x0000010c	Sharpness setting value
0x0000010d	Color space setting
0x0000010e	Tone curve (standard or custom)
0x0000010f	Color effect setting
0x00000110	Filter effect setting
0x00000111	Gradation effect setting
0x00000112	Processing parameter setting
0x00000113	Color matrix setting
0x00000114	Picture style
0x00000115	Picture style setting details
0x00000200	Computer settings caption for the picture style at the time of shooting

■ Develop Properties

Value	Description
0x00000300	Linear processing ON/OFF
0x00000301	Click WB coordinates
0x00000302	WB control value

■ Capture Properties

Value	Description
0x00000400	Shooting mode
0x00000401	Drive mode (cap)
0x00000402	ISO sensitivity setting value
0x00000403	Metering mode
0x00000404	AF mode (cap)
0x00000405	Aperture value (cap) at the time of shooting
0x00000406	Shutter speed setting value (cap)
0x00000407	Exposure compensation (cap)
0x00000408	Flash compensation setting
0x00000409	Lens focal length information at the time of shooting
0x0000040a	Number of available shots
0x0000040b	ISO, auto exposure or flash exposure bracket
0x0000040c	White balance bracket
0x0000040d	String representing the lens name
0x0000040e	Auto exposure bracket value
0x0000040f	Flash exposure bracket value
0x00000410	ISO bracket value
0x00000411	Noise reduction
0x00000412	Use of the flash (activated or not)
0x00000413	Red-eye reduction
0x00000414	Flash type

■ Other

Value	Description
0x0000FFFF	Unknown

Revision History/Date		Corrections	Reviser	Remarks

5.2 Property Details

Properties are explained in the following format.

5.3.xx PropertyID

The property ID.

Description

Explains the role of the property and how to work with it.

Target Object

Indicates the "target object" that the property describes and which is subject to operations involving the property.
Properties for which "Access Type" is [Read] can be read by means of objects subject to operations, such as remote cameras. Similarly, an access type of [Write] means the property can be set by means of operations on objects subject to operations.
"Data type number" indicates the enumeration name for data types that can be retrieved by means of **EdsGetPropertySize**.
"Data type" indicates the data type of property data that can be retrieved or set by means of an **EdsVoid** pointer, which is a dummy argument for **EdsGetPropertyData** or **EdsSetPropertyData**.

Value

Indicates possible values for the property.
Values are expressed as decimals unless otherwise noted.

Note

Considerations when using the property.

5.2.1 kEdsPropID_AtCapture_Flag

Description

A supporting property for getting the properties at the time of shooting.
This property ID cannot be used by itself.

Usually, the properties you can retrieve from objects are the current settings values. However, if the target object is EdsImageRef, when getting image properties, you can get some properties at the time of shooting by designating a property ID and an OR value for kEdsPropID_AtCapture_Flag in the arguments for **EdsGetPropertyData**.

The property types of values at the time of shooting that can be retrieved are as follows.

Properties that can be retrieved for settings values at the time of shooting
kEdsPropID_DigitalExposure
kEdsPropID_WhiteBalance
kEdsPropID_ColorTemperature
kEdsPropID_WhiteBalanceShift
kEdsPropID_ClickWBPoint
kEdsPropID_WBCoeffs
kEdsPropID_Linear

Revision History/Date		Corrections	Reviser	Remarks

kEdsPropID_Sharpness
kEdsPropID_ColorMatrix
kEdsPropID_ColorSaturation
kEdsPropID_Contrast
kEdsPropID_ColorTone
kEdsPropID_ColorSpace
kEdsPropID_PhotoEffect
kEdsPropID_FilterEffect
kEdsPropID_ToningEffect
kEdsPropID_PictureStyle
kEdsPropID_PictureStyleDesc

Target Object

Target object	Access type	Data type number	Data type
EdsImageRef	Read	kEdsDataType_UInt32	EdsUInt32

Value

None

5.2.2 kEdsPropID_ProductName

Description

A string representing the product name.
If the target object is EdsCameraRef, this property indicates the name of the remote camera.
If the target object is EdsImageRef, this property indicates the name of the camera used to shoot the image.

Data Type

Data type number	Data type
kEdsDataType_String	EdsChar[]

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read	kEdsDataType_String	EdsChar[]
EdsImageRef			

Value

ASCII text strings up to 32 characters, including null-terminated strings.

5.2.3 kEdsPropID_BodyID

Description

Indicates the product serial number.
If the target object is EdsCameraRef, this property indicates the serial number of the remote camera.
If the target object is EdsImageRef, this property indicates the serial number of the camera used to shoot the image.

Data Type

Revision History/Date		Corrections	Reviser	Remarks

Data type number	Data type
kEdsDataType_UInt32	EdsUInt32

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read	kEdsDataType_UInt32	EdsUInt32
EdsImageRef			

Value

Integer values.

5.2.4 kEdsPropID_OwnerName

Description

Indicates a string identifying the owner as registered on the camera.
If the target object is EdsCameraRef, this property indicates the owner name for the remote camera.
If the target object is EdsImageRef, this property indicates the owner name for the camera used to shoot the image.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_String	EdsChar[]
EdsImageRef	Read	kEdsDataType_String	EdsChar[]

Value

ASCII text strings up to 32 characters, including null-terminated strings.

5.2.5 kEdsPropID_MakerName

Description

Indicates a string identifying the manufacturer.

Target Object

Target object	Access type	Data type number	Data type
EdsImageRef	Read	kEdsDataType_String	EdsChar[]

Value

ASCII strings, including null-terminated strings. For our purposes: "Canon".

5.2.6 kEdsPropID_DateTime

Description

Indicates the time and date set on the camera or the shooting date and time of images.
If the target object is EdsCameraRef, this property indicates the camera system time.
If the target object is EdsImageRef, this property indicates the time and date of shooting.

Target Object

Target object	Access type

Revision History/Date		Corrections	Reviser	Remarks

EdsCameraRef	Read
EdsImageRef	Read

Value

The time and date as an EdsTime type; for Read or Write operations.

5.2.7 kEdsPropID_FirmwareVersion

Description

Indicates the camera's firmware version.

Data Type

Data type number	Data type
kEdsDataType_String	EdsChar[]

Target Object

Target object	Access type
EdsCameraRef	Read
EdsImageRef	Read

Value

ASCII text strings up to 32 characters, including null-terminated strings.

5.2.8 kEdsPropID_BatteryLevel

Description

Indicates the camera battery level.
When the battery reaches a particular level, a kEdsPropertyEvent_PropertyChanged event is generated. The battery level that triggers the event is model-dependent.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read	kEdsDataType_UInt32	EdsUInt32

Value

Value	Description
0–100	Battery level (%)
0xffffffff	AC power

5.2.9 kEdsPropID_CFn

Description

Indicates the state of the camera's Custom Function setting.
When retrieving properties by means of EdsGetPropertyData, designate the Custom Function number in the inParam argument and identify the number of the Custom Function for which you seek settings values.
The meaning of the Custom Function number varies depending on the camera model. For details on the relationship of the number to the Custom Function, see the camera's Custom Function (C.Fn).

Revision History/Date		Corrections	Reviser	Remarks

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read	kEdsDataType_UInt32	EdsUInt32
EdsImageRef			

Value

The setting value of the Custom Function designated in inParam. The possible values vary depending on the Custom Function.
For details, see the camera user's manual.

Note

- There is no interface for retrieving settings values for all Custom Functions at once.
- Only Read is allowed as the access type. These values cannot be set on cameras from a host computer.

5.2.10 kEdsPropID_PFn

Description

Indicates the state of the camera's Personal Function setting and its settings values.
When retrieving properties by means of EdsGetPropertyData, designate the Personal Function number in the inParam argument and identify the number of the Personal Function for which you seek settings values.
The meaning of the Personal Function number varies depending on the camera model. For details on the relationship of the number to the Personal Function, see the camera Personal Function (P.Fn).

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/(Write)	kEdsDataType_Int32_Array	EdsInt32[]
EdsImageRef	Read		

Value

The setting value of the Personal Function designated in inParam. The value varies depending on the Personal Function.
Because erroneous property value settings may break the camera, property value are not disclosed.

Note

- There is no interface for retrieving settings values for all Personal Functions at once.
- Unlike Custom Functions, Personal Functions can be set on cameras from a host computer. However, setting the wrong property value may break the camera, so avoid setting (that is, using Write for) this property.

5.2.11 kEdsPropID_FocusInfo

Description

Indicates focus information for image data at the time of shooting.
This property does not depend on the AF mode at the time of shooting. AF frames in focus are indicated by JustFocus, even during manual shooting.

Target Object

Target object	Access type	Data type number	Data type

Revision History/Date		Corrections	Reviser	Remarks

EdsImageRef	Read	kEdsDataType_FocusInfo	EdsFocusInfo

Value

Element		Value
imageRect		The upper-left coordinates of the image, as well as the width and height
pointNumber		AF frame number
focusPoint	valid	Invalid AF frame: 0 Valid AF frame: 1 Note: There are as many valid AF frames as the number in FrameNumber. Usually, AF frames are recorded consecutively, starting with 0. Note: AF frame coordinates and the array number for storage vary by model.
	justFocus	In focus: 1 Out of focus: 0
	rect	Upper-left and lower-right coordinates of the AF frame
	reserved	Reserved

5.2.12 kEdsPropID_ICCProfile

Description

Indicates the ICC profile data embedded in an image.
An error is returned if you use EdsGetPropertyData to attempt to get the ICC profile of an image without an embedded ICC profile.

Target Object

Target object	Access type	Data type number	Data type
EdsImageRef	Read	kEdsDataType_ByteBlock	EdsInt8[]

Value

Returns ICC profile data as ByteBlock data.

5.2.13 kEdsPropID_ImageQuality

Description

Indicates the image quality.
If you designate EdsCameraRef as the target object, this property indicates the current image quality set on the camera.
If you designate an image as the target object, this property indicates the image quality that the image was shot with.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_UInt32_Array	EdsUInt32[]
EdsImageRef	Read		

Value

Array	Description	Value

Revision History/Date		Corrections	Reviser	Remarks

number		
0	Image Size of the main image	Values defined in enum EdsImageSize
1	Image Type of the main image	Values defined in enum EdsImageType
2	Image Quality of the main image	Values defined in enum EdsImageQuality
3	Image Size of the secondary image	Values defined in enum EdsImageSize
4	Image Type of the secondary image	Values defined in enum EdsImageType
5	Image Quality of the secondary image	Values defined in enum EdsImageQuality

- If your target object is EdsCameraRef, "main" and "secondary" images refers to RAW and JPEG images, respectively, when shooting in RAW+JPEG mode. When shooting in RAW or JPEG mode, there is no secondary image, so it is indicated as UNKNOWN.
- If your target object is EdsImage, there is no secondary image, so the image quality indicated for secondary images is UNKNOWN. Main images of RAW images are in RAW image quality; main images of JPEG images are in JPEG image quality.
- For camera models that do not support size selection for RAW images at the time of shooting, the Image Size of main images is UNKNOWN.

EdsImageType <defined location>EDSDKTypes.h

Value	Description
kEdsTargetImageType _Unknown	Folder, or unknown image type
kEdsTargetImageType _Jpeg	JPEG
kEdsImageType_CRW	CRW
kEdsImageType_CR2	CR2

EdsImageSize <defined location>EDSDKTypes.h

Value	Description
0	Large
1	Medium
2	Small
5	Medium 1
6	Medium 2
0xFFFFFFFF	Unknown

EdsImageQuality <defined location>EDSDKTypes.h

Value	Description
2	Normal
3	Fine
4	Lossless
5	Superfine
0xFFFFFFFF	Unknown

5.2.14 kEdsPropID_JpegQuality

Description

Revision History/Date		Corrections	Reviser	Remarks

Indicates the JPEG compression.
In the inParam argument, designate Image Size as retrieved by means of the kEdsPropID_ImageQuality property.
This property is valid for the EOS 1 series only.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	KEdsDataType_UInt32	EdsUInt32
EdsImageRef	Read		

Value

An integer value of 0–10. (0 if uncompressed.)

5.2.15 kEdsPropID_Orientation

Description

Indicates image rotation information.
This property can be read or written, regardless of the image compression format (RAW, JPEG, and so on); the access type is Read/Write.

Target Object

Target object	Access type	Data type number	Data type
EdsImageRef	Read/Write	kEdsDataType_UInt32	EdsUInt32

Value

Value	Description	U: Up D: Down L: Left R: Right
1	The 0th row is at the visual top of the image, and the 0th column is on the visual left-hand side	U L + R D
3	The 0th row is at the visual bottom of the image, and the 0th column is on the visual right-hand side	D R + L U
6	The 0th row is on the visual right-hand side of the image, and the 0th column is at the visual top	L D + U R
8	The 0th row is on the visual left-hand side of the image, and the 0th column is at the visual bottom	R U + D L
Other	Reserved	

Note

Rotation information is retrieved from images' Exif information. Thus, images rotated by means of a software tool of computer may be displayed differently from how they would appear using the actual rotation information.

Revision History/Date		Corrections	Reviser	Remarks

5.2.16 kEdsPropID_AEMode

Description

Indicates settings values of the camera in shooting mode.
You cannot set (that is, Write) this property on cameras with a mode dial.
If the target object is EdsCameraRef, you can use GetPropertyDesc to access this property and get a list of property values that can currently be set.
However, you cannot get a list of settable values from models featuring a dial. The GetPropertyDesc return value will be EDS_ERR_OK, and no items will be listed as values you can set.
The shooting mode is in either an applied or simple shooting zone. When a camera is in a shooting mode of the simple shooting zone, a variety of capture-related properties (such as for auto focus, drive mode, and metering mode) are automatically set to the optimal values. Thus, when the camera is in a shooting mode of a simple shooting zone, capture-related properties cannot be set on the camera.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/(Write)	kEdsDataType_UInt32	EdsUInt32
EdsImageRef	Read		

Value

Values defined in Enum EdsAEMode.

Enum EdsAEMode

Value		Description
0	Applied shooting zone	Program AE
1		Shutter-Speed Priority AE
2		Aperture Priority AE
3		Manual Exposure
4		Bulb Note: For some models, the value of the property cannot be retrieved as kEdsPropID_AEMode. In this case, Bulb is retrieved as the value of the shutter speed (kEdsPropID_Tv) property. Note: Bulb is designed so that it cannot be set on cameras from a computer by means of SetPropertyData.
5		Auto Depth-of-Field AE
6		Depth-of-Field AE
7		Camera settings registered
8		Lock
9	Simple shooting	Auto
10		Night Scene Portrait
11		Sports
12		Portrait
13		Landscape
14		Close-Up
15		Flash Off
0xFFFFFFFF		Not valid/no settings changes

5.2.17 kEdsPropID_DriveMode

Description

Indicates settings values of the camera in drive mode.

Revision History/Date		Corrections	Reviser	Remarks

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_Uint32	EdsUInt32
EdsImageRef	Read		

Value

Value	Description
0x00000000	Single-Frame Shooting
0x00000001	Continuous Shooting
0x00000002	Video
0x00000003	Not used
0x00000004	High-Speed Continuous Shooting
0x00000005	Low-Speed Continuous Shooting
0x00000010	10-Sec Self-Timer
0x00000011	2-Sec Self-Timer

5.2.18 kEdsPropID_ISOSpeed

Description

Indicates ISO sensitivity settings values.
Caution is advised because it is possible to retrieve different values by means of EdsCameraRef and EdsImageRef.
If the target object is EdsCameraRef, you can use GetPropertyDesc to access this property and get a list of property values that can currently be set.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_Uint32	EdsUInt32
EdsImageRef	Read		

Value (EdsCameraRef)

Value	Description
0x00000028	ISO 6
0x00000030	ISO 12
0x00000038	ISO 25
0x00000040	ISO 50
0x00000048	ISO 100
0x0000004b	ISO 125
0x0000004d	ISO 160
0x00000050	ISO 200
0x00000053	ISO 250
0x00000055	ISO 320
0x00000058	ISO 400
0x0000005b	ISO 500
0x0000005d	ISO 640
0x00000060	ISO 800
0x00000063	ISO 1000
0x00000065	ISO 1250

Revision History/Date		Corrections	Reviser	Remarks

0x00000068	ISO 1600
0x00000070	ISO 3200
0x00000078	ISO 6400
0xffffffff	Not valid/no settings changes

Value (EdsImageRef)

Value	Description
50	ISO 50
100	ISO 100
200	ISO 200
400	ISO 400
800	ISO 800
1600	ISO 1600
3200	ISO 3200

The value you can retrieve from the image data, indicated by EdsImageRef, represents the ISO value itself.

5.2.19 kEdsPropID_MeteringMode

Description

Indicates the metering mode.
If the target object is EdsCameraRef, you can use GetPropertyDesc to access this property and get a list of property values that can currently be set.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_Uint32	EdsUInt32
EdsImageRef	Read		

Value

Value	Description
1	Spot metering
3	Evaluative metering
4	Partial metering
5	Center-weighted averaging metering
0xFFFFFFFF	Not valid/no settings changes

Note

For details on various metering modes, see the camera user's manual.

5.2.20 kEdsPropID_AFMode

Description

Indicates AF mode settings values.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read	kEdsDataType_Uint32	EdsUInt32

Revision History/Date		Corrections	Reviser	Remarks

EdsImageRef			

Value

Value	Description
0	One-Shot AF
1	AI Servo AF
2	AI Focus AF
3	Manual Focus
0xffffffff	Not valid/no settings changes

5.2.21 kEdsPropID_Av

Description

Indicates the camera's aperture value.
Caution is advised because EdsCameraRef and EdsImageRef yield different data types and values.
If the target object is EdsCameraRef, you can use GetPropertyDesc to access this property and get a list of property values that can currently be set.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_Uint32	EdsUInt32
EdsImageRef	Read	kEdsType_Rational	EdsRational

Value (EdsCameraRef)

Value	Aperture value
0x08	1
0x0B	1.1
0x0C	1.2
0x0D	1.2 (1/3)
0x10	1.4
0x13	1.6
0x14	1.8
0x15	1.8 (1/3)
0x18	2
0x1B	2.2
0x1C	2.5
0x1D	2.5 (1/3)
0x20	2.8
0x23	3.2
0x24	3.5
0x25	3.5 (1/3)
0x28	4
0x2B	4 (1/3)
0x2C	4.5
0x2D	5.6 (1/3)
0x30	5.6
0x33	6.3
0x34	6.7
0x35	7.1

Value	Aperture value
0x40	11
0x43	13 (1/3)
0x44	13
0x45	14
0x48	16
0x4B	18
0x4C	19
0x4D	20
0x50	22
0x53	25
0x54	27
0x55	29
0x58	32
0x5B	36
0x5C	38
0x5D	40
0x60	45
0x63	51
0x64	54
0x65	57
0x68	64
0x6B	72
0x6C	76
0x6D	80

Revision History/Date		Corrections	Reviser	Remarks

0x38	8
0x3B	9
0x3C	9.5
0x3D	10

0x70	91
0xffffffff	Not valid/no settings changes

Note: Values labeled "(1/3)" represent property values when the step set in the Custom Function is 1/3.

Value (EdsImageRef)

Returns the aperture value as an EdsRational type.

5.2.22 kEdsPropID_Tv

Description

Indicates the shutter speed.
Caution is advised because EdsCameraRef and EdsImageRef yield different values.
If the target object is EdsCameraRef, you can use GetPropertyDesc to access this property and get a list of property values that can currently be set.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_Uint32	EdsUInt32
EdsImageRef	Read	kEdsType_Rational	EdsRational

Value (EdsCameraRef)

Value	Shutter speed
0x0C	Bulb
0x10	30"
0x13	25"
0x14	20"
0x15	20" (1/3)
0x18	15"
0x1B	13"
0x1C	10"
0x1D	10" (1/3)
0x20	8"
0x23	6" (1/3)
0x24	6"
0x25	5"
0x28	4"
0x2B	3"2
0x2C	3"
0x2D	2"5
0x30	2"
0x33	1"6
0x34	1"5
0x35	1"3
0x38	1"
0x3B	0"8
0x3C	0"7

Value	Shutter speed
0x5D	1/25
0x60	1/30
0x63	1/40
0x64	1/45
0x65	1/50
0x68	1/60
0x6B	1/80
0x6C	1/90
0x6D	1/100
0x70	1/125
0x73	1/160
0x74	1/180
0x75	1/200
0x78	1/250
0x7B	1/320
0x7C	1/350
0x7D	1/400
0x80	1/500
0x83	1/640
0x84	1/750
0x85	1/800
0x88	1/1000
0x8B	1/1250
0x8C	1/1500

Revision History/Date		Corrections	Reviser	Remarks

0x3D	0"6
0x40	0"5
0x43	0"4
0x44	0"3
0x45	0"3 (1/3)
0x48	1/4
0x4B	1/5
0x4C	1/6
0x4D	1/6 (1/3)
0x50	1/8
0x53	1/10 (1/3)
0x54	1/10
0x55	1/13
0x58	1/15
0x5B	1/20 (1/3)
0x5C	1/20

0x8D	1/1600
0x90	1/2000
0x93	1/2500
0x94	1/3000
0x95	1/3200
0x98	1/4000
0x9B	1/5000
0x9C	1/6000
0x9D	1/6400
0xA0	1/8000
0xffffffff	Not valid/no settings changes

Note: Values labeled "(1/3)" represent property values when the step set in the Custom Function is 1/3.

Value (EdsImageRef)

Returns the shutter speed value as a kEdsType_Rational type.

Note

- Bulb is designed so that it cannot be set on cameras from a computer by means of SetPropertyData. (It cannot even be retrieved by means of GetPropertyDesc as a value that can be set.) This is because incorrect handling of Bulb would prevent shutter control from a computer.

5.2.23 kEdsPropID_ExposureCompensation

Description

Indicates the exposure compensation.
Exposure compensation refers to compensation relative to the position of the standard exposure mark (in the center of the exposure gauge).
Caution is advised because EdsCameraRef and EdsImageRef yield different values.
If the target object is EdsCameraRef, you can use GetPropertyDesc to access this property and get a list of property values that can currently be set.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_Uint32	EdsUInt32
EdsImageRef	Read	kEdsType_Rational	EdsRational

Value (EdsCameraRef)

Value	Exposure compensation
0x18	+3
0x15	+2 2/3
0x14	+2 1/2
0x13	+2 1/3
0x10	+2

Value	Exposure compensation
0xFD	–1/3
0xFC	–1/2
0xFB	–2/3
0xF8	–1
0xF5	–1 1/3

Revision History/Date		Corrections	Reviser	Remarks

0x0D	+1 2/3
0x0C	+1 1/2
0x0B	+1 1/3
0x08	+1
0x05	+2/3
0x04	+1/2
0x03	+1/3
0x00	0

0xF4	–1 1/2
0xF3	–1 2/3
0xF0	–2
0xED	–2 1/3
0xEC	–2 1/2
0xEB	–2 2/3
0xE8	–3
0xffffffff	Not valid/no settings changes

Value (EdsImageRef)

Returns the exposure compensation as a kEdsType_Rational type.

Note

• Exposure compensation is not available if the camera is in manual exposure mode. Thus, the exposure compensation property is invalid.

5.2.24 kEdsPropID_DigitalExposure

Description

Indicates the digital exposure compensation.
As the digital exposure compensation, a value is returned representing the compensation for brightness. This is equivalent to the exposure at the time of shooting as adjusted for the aperture plus or minus several steps.

Target Object

Target object	Access type	Data type number	Data type
EdsImageRef	Read/Write	kEdsType_Rational	EdsRational

Note: Write is available as the access type with EdsImageRef only for RAW images.

Value

Returns the digital exposure compensation as a kEdsType_Rational type.

See Also

• Regarding RAW support for each camera model, to determine if a property is valid during processing, see Support Status for RAW Properties.

Note

• With this property, it is possible to get values at the time of shooting.

5.2.25 kEdsPropID_FlashCompensation

Description

Indicates the flash compensation.
Note that flash compensation cannot be retrieved for an external flash.

Target Object

Revision History/Date		Corrections	Reviser	Remarks

Target object	Access type	Data type number	Data type
EdsCameraRef	Read	kEdsDataType_Uint32	EdsUInt32

Value

The flash compensation is the same value as the exposure compensation property **kEdsPropID_ExposureCompensation**.

5.2.26 kEdsPropID_FocalLength

Description

Indicates the focal length of the lens.
When a single-focus lens is used, the same value is returned for the Wide and Tele focal length.
You can obtain three items of information at once by using EdsGetPropertyData to get this property: the focal length at the time of shooting, the focal length of Wide, and the focal length of Tele. In this case, the buffer storing this property data is passed in three parts. However, if you prefer to get only the focal length at the time of shooting, you can get only that single part of the buffer.

Example: To get only the focal length at the time of shooting

```
EdsRatioal ratVal ;
err = EdsGetPropertyData( ref, kEdsPropID_FocalLength, 0, sizeof( EdsRational ), &ratVal ) ;
```

Target Object

Target object	Access type	Data type number	Data type
EdsImageRef	Read	kEdsDataType_Rational_Array	EdsRational[]

Value

Array number	Description	Value
0	Focal length at the time of shooting	Focal length value
1	Wide focal length	
2	Tele focal length	

5.2.27 kEdsPropID_AvailableShots

Description

Indicates the number of shots available on a camera.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read	kEdsDataType_Uint32	EdsUInt32

Value

Integer values.

Note

- Remote type 2 protocol standard cameras return the number of shots left on the camera based on the available disk capacity of the host computer they are connected to.

Revision History/Date		Corrections	Reviser	Remarks

5.2.28 kEdsPropID_Bracket

Description

Indicates the current bracket type.
If multiple brackets have been set on the camera, you can get the bracket type as a logical sum.
This property cannot be used to get bracket compensation. Compensation is collected separately because there are separate properties for each bracket type.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef EdsImageRef	Read	kEdsDataType_Uint32	EdsUInt32

Value

Values defined in Enum EdsBracket.

Enum EdsBracket <defined location>EDSDKType.h

Value	Description
0x01	AE bracket
0x02	ISO bracket
0x04	WB bracket
0x08	FE bracket
0xFFFFFFFF	Bracket off

5.2.29 kEdsPropID_AEBracket

Description

Indicates the AE bracket compensation of image data.

Target Object

Target object	Access type	Data type number	Data type
EdsImageRef	Read	kEdsDataType_Rational	EdsRational

Value

Returns the AE bracket compensation. For details on the compensation range and number of steps, see the camera user's manual.

5.2.30 kEdsPropID_FEBracket

Description

Indicates the FE bracket compensation at the time of shooting of image data.

Target Object

Target object	Access type	Data type number	Data type
EdsImageRef	Read	kEdsDataType_Rational	EdsRational

Value

Returns the FE bracket compensation. For details on the compensation range and number of steps, see the

Revision History/Date		Corrections	Reviser	Remarks

camera user's manual.

5.2.31 kEdsPropID_ISOBracket

Description

Indicates the ISO bracket compensation at the time of shooting of image data.

Target Object

Target object	Access type	Data type number	Data type
EdsImageRef	Read	kEdsDataType_Rational	EdsRational

Value

Returns the ISO bracket compensation. For details on the compensation range and number of steps, see the camera user's manual.

5.2.32 kEdsPropID_ WhiteBalanceBracket

Description

Indicates the white balance bracket amount.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read	kEdsDataType_Int32_Array	EdsInt32[]
EdsImageRef			

Value(EdsCameraRef)

Array number	Description	Value
0	BracketMode	0 = OFF 1 = Mode AB 2 = Mode GM 0xFFFFFFFF = Not Supported
1	BracketValueAB The bracket amount from the WhiteBalanceShift position toward AB	0 to +9
2	BracketValueGM The bracket amount from the WhiteBalanceShift position toward GM	0 to +9

Note: "AB" means the bracket toward amber-blue and "GM" toward green-magenta.

Note

- Under the camera specifications, AB and GM modes cannot be set at the same time.
- Depending on the model, it may not be possible to get an accurate value.
  For example, no value is specified in BracketMode for the EOS Kiss Digital N/350D/REBEL XT, and 3 is specified in BracketValueAB regardless of the bracket amount. (It can be known that the camera's WB bracket has been set.)

Value (EdsImageRef)

Array	Description	Value

Revision History/Date		Corrections	Reviser	Remarks

number		
0	BracketMode	0 = OFF 1 = Mode AB 2 = Mode GM 0xFFFFFFFF = Not Supported
1	BracketValueAB The bracket amount from the WhiteBalanceShift position toward AB	–9 to +9 (B direction–A direction)
2	BracketValueGM The bracket amount from the WhiteBalanceShift position toward GM	–9 to +9 (G direction–M direction)

5.2.33 kEdsPropID_WhiteBalance

Description

Indicates the white balance type.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_Int32	EdsInt32
EdsImageRef	Read/Write		

Note: Write is available as the access type with EdsImageRef only for RAW images.

Value

Values defined in Enum EdsWhiteBalance.

Enum EdsWhiteBalance <defined location>EDSDKType.h

Value	Description
0	Auto
1	Daylight
2	Cloudy
3	Tungsten
4	Fluorescent
5	Flash
6	Manual (set by shooting a white card or paper)
8	Shade
9	Color temperature
10	Custom white balance: PC-1
11	Custom white balance: PC-2
12	Custom white balance: PC-3
–1	Setting the white balance by clicking image coordinates
–2	White balance copied from another image

See Also

- Regarding RAW support for each camera model, to determine if a property is valid during processing, see Support Status for RAW Properties.

Note

- If the white balance type is "Color Temperature," to know the actual color temperature you must reference another property (kEdsPropID_ColorTemperature).

Revision History/Date		Corrections	Reviser	Remarks

• With this property, it is possible to get values at the time of shooting.

5.2.34 kEdsPropID_ColorTemperature

Description

Indicates the color temperature setting value. (Units: Kelvin)
Valid only when the white balance is set to Color Temperature.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_UInt32	EdsUInt32
EdsImageRef	Read/Write		

Note: Write is available as the access type with EdsImageRef only for RAW images.

Value

2800–10000, in 100-Kelvin increments.
5200 represents a color temperature of 5200 K.

See Also

• Regarding RAW support for each camera model, to determine if a property is valid during processing, see Support Status for RAW Properties.

Note

• To know if the white balance is set to color temperature, refer to another property (kEdsPropID_WhiteBalance).
• With this property, it is possible to get values at the time of shooting.

5.2.35 kEdsPropID_WhiteBalanceShift

Description

Indicates the white balance compensation.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_Int32_Array	EdsInt32[]
EdsImageRef	Read/Write		

Note: Write is available as the access type with EdsImageRef only for RAW images.

Value

Array number	Description	Value
0	ValueAB	–9 to +9 0x7FFFFFFF = invalid value Note: 0 means no compensation, (–) means compensation toward blue, and (+) means compensation toward amber.
1	ValueGM	–9 to +9 0x7FFFFFFF = invalid value Note: 0 means no compensation, (–) means compensation toward green, and (+) means

Revision History/Date		Corrections	Reviser	Remarks

		compensation toward magenta.

Note: "AB" means compensation toward amber-blue and "GM" toward green-magenta.

See Also

- Regarding RAW support for each camera model, to determine if a property is valid during processing, see Support Status for RAW Properties.

Note

- With this property, it is possible to get values at the time of shooting.

5.2.36 kEdsPropID_ClickWBPoint

Description

Indicates the coordinates when an image is clicked to set the white balance.
Only writing is valid.
If you designate coordinates for this property, the white balance value for those coordinates is incorporated in the property kEdsPropID_WBCoeffs.

Target Object

Target object	Access type	Data type number	Data type
EdsImageRef	Write	kEdsDataType_Point	EdsPoint

Value

Designate coordinates within the range of the target image.

Note

- With this property, it is possible to get values at the time of shooting.

5.2.37 kEdsPropID_WBCoeffs

Description

Indicates the white balance value.
You can apply this value to other image properties, to process images under the same white balance.

Target Object

Target object	Access type	Data type number	Data type
EdsImageRef	Read/Write	kEdsDataType_ByteBlock	EdsInt8[]

Note: Write is available as the access type with EdsImageRef only for RAW images.

Value

Coefficients to maintain a specific white balance. Use unmodified data from a source image with a white balance you want to copy.

See Also

- Regarding RAW support for each camera model, to determine if a property is valid during processing, see Support Status for RAW Properties.

Note

- With this property, it is possible to get values at the time of shooting.

Revision History/Date		Corrections	Reviser	Remarks

5.2.38 kEdsPropID_Linear

Description

Indicates if linear processing is activated or not.
This property is valid only if 16-bit TIFF or 16-bit RGB has been set for image processing.

Target Object

Target object	Access type	Data type number	Data type
EdsImageRef	Read/Write	kEdsDataType_Bool	EdsBool

Note: Write is available as the access type with EdsImageRef only for RAW images.

Value

TRUE: Linear processing
FALSE: Not linear processing

Note

• With this property, it is possible to get values at the time of shooting.

5.2.39 kEdsPropID_Sharpness

Description

Indicates the sharpness setting.
If the target object is EdsCameraRef and you designate the processing parameter set (refer to the kEdsPropID_ParameterSet value) in inParam, this property corresponds to the sharpness setting value of that processing parameter set. By using inParam = 0, you can designate the current sharpness.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_Int32	EdsInt32
EdsImageRef (Other than 1D/1Ds)	Read		
EdsImageRef (1D/1Ds)	Read	kEdsDataType_Int32_Array	EdsInt32[]

Note: Write is available as the access type with EdsImageRef only for RAW images.

Value (EdsCameraRef and EdsImageRef for models other than 1D and 1Ds)

Value	Description
0 to 5	1 series models
–2 to 2	20D, Kiss Digital N/350D/REBEL XT
0x7FFFFFFF	Unknown

Value (EdsImageRef, 1D and 1Ds)

Array number	Description	Value
0	Sharpness	0: Invalid 1 2 3 4 5 Weaker <————> Stronger
1	Applicable sharpness	0 1 2 3 4 5

Revision History/Date		Corrections	Reviser	Remarks

		Rough <————————> Fine

See Also

- Regarding RAW support for each camera model, to determine if a property is valid during processing, see Support Status for RAW Properties.

Note

- This property cannot be retrieved or set from EdsCameraRef for the EOS 20D or EOS Kiss Digital N/350D/REBEL XT.
- This property is invalid for models supporting picture styles. For models supporting picture styles, use the property kEdsPropID_PictureStyleDesc.
- With this property, it is possible to get values at the time of shooting.

5.2.40 kEdsPropID_ParameterSet

Description

Indicates the current processing parameter set on a camera.
Only valid for the EOS 1D Mark II and EOS 1Ds Mark II.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_UInt32	EdsUInt32

Value

Value	Description
0	Standard (Read only)
1	Processing parameter 1
2	Processing parameter 2
3	Processing parameter 3

5.2.41 kEdsPropID_ColorSaturation

Description

Indicates the saturation.
If the target object is EdsCameraRef and you designate ColorMatrix in inParam, this property corresponds to the saturation setting value of ColorMatrix. By using inParam = 0, you can designate the current saturation value.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_Int32	EdsInt32
EdsImageRef	Read/Write		

Note: Write is available as the access type with EdsImageRef only for RAW images.

Value

Value	Description
–2 to 2	For the 20D, Kiss Digital N/350D/REBEL XT, 1D Mark II, or 1Ds Mark II
0x7FFFFFFF	Unknown

Revision History/Date		Corrections	Reviser	Remarks

See Also

- Regarding RAW support for each camera model, to determine if a property is valid during processing, see Support Status for RAW Properties.

Note

- This property cannot be retrieved or set from EdsCameraRef for the 20D or Kiss Digital N/350D/REBEL XT.
- This property is invalid for models supporting picture styles. For models supporting picture styles, use the property kEdsPropID_PictureStyleDesc.
- With this property, it is possible to get values at the time of shooting.

5.2.42 kEdsPropID_ColorMatrix

Description

Indicates the color matrix.
Only valid for the EOS 1D Mark II and EOS 1Ds Mark II.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_UInt32	EdsUInt32
EdsImageRef	Read/Write		

Note: Write is available as the access type with EdsImageRef only for RAW images.

Value

Values defined in Enum EdsColorMatrix.

Enum EdsColorMatrix <defined location>EDSDKTypes.h

Value	Description
1	ColorMatrix1
2	ColorMatrix2
3	ColorMatrix3
4	ColorMatrix4
5	ColorMatrix5
6	ColorMatrix6
7	ColorMatrix7
0x7FFFFFFF	Unknown Note: "Unknown" also applies for a color matrix customized on a computer and set on the camera.

See Also

- Regarding RAW support for each camera model, to determine if a property is valid during processing, see Support Status for RAW Properties.

Note

- With this property, it is possible to get values at the time of shooting.

Revision History/Date		Corrections	Reviser	Remarks

5.2.43 kEdsPropID_Contrast

Description

Indicates the contrast.
If the target object is EdsCameraRef and you designate the processing parameter set in inParam, this property corresponds to that setting value. By using inParam = 0, you can designate the current contrast value.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_Int32	EdsInt32
EdsImageRef	Read/Write		

Note: Write is available as the access type with EdsImageRef only for RAW images.

Value

Value	Description
–2 to 2	20D, Kiss Digital N/350D/REBEL XT, 1D Mark II, 1Ds Mark II
0x7FFFFFFF	Unknown

See Also

- Regarding RAW support for each camera model, to determine if a property is valid during processing, see Support Status for RAW Properties.

Note

- This property cannot be retrieved or set from EdsCameraRef for the 20D or Kiss Digital N/350D/REBEL XT.
- This property is invalid for models supporting picture styles. For models supporting picture styles, use the property kEdsPropID_PictureStyleDesc.
- With this property, it is possible to get values at the time of shooting.

5.2.44 kEdsPropID_ColorTone

Description

Indicates the color tone.
If the target object is EdsCameraRef and you designate ColorMatrix in inParam, this property corresponds to the color tone setting value of ColorMatrix. Similarly, if you designate the processing parameter in inParam, it indicates the color tone setting value of the item you designated. By using inParam = 0, you can designate the current color tone.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_Int32	EdsInt32
EdsImageRef	Read/Write		

Note: Write is available as the access type with EdsImageRef only for RAW images.

Value

Value	Description
–2 to 2	20D, Kiss Digital N/350D/REBEL XT, 1D Mark II, 1Ds Mark II
0x7fffffff	Unknown

See Also

Revision History/Date		Corrections	Reviser	Remarks

• Regarding RAW support for each camera model, to determine if a property is valid during processing, see Support Status for RAW Properties.

Note

• This property cannot be retrieved or set from EdsCameraRef for the 20D or Kiss Digital N/350D/REBEL XT.
• This property is invalid for models supporting picture styles. For models supporting picture styles, use the property kEdsPropID_PictureStyleDesc.
• With this property, it is possible to get values at the time of shooting.

5.2.45 kEdsPropID_ColorSpace

Description

Indicates the color space.
If the target object is EdsCameraRef and you designate ColorMatrix in inParam, this property corresponds to the color space setting value of ColorMatrix. Similarly, if you designate the processing parameter in inParam, it indicates that setting value. By using inParam = 0, you can designate the current color space.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_UInt32	EdsUInt32
EdsImageRef	Read/Write		

Note: Write is available as the access type with EdsImageRef only for RAW images.

Value

Values of Enum EdsColorSpace.

Enum EdsColorSpace <defined location>EDSDKTypes.h

Value	Description
1	sRGB
2	Adobe RGB
0xFFFFFFFF	Unknown

See Also

• Regarding RAW support for each camera model, to determine if a property is valid during processing, see Support Status for RAW Properties.

Note

• With this property, it is possible to get values at the time of shooting.

5.2.46 kEdsPropID_PhotoEffect

Description

Indicates the photo effect.
This property is valid only for the 20D and Kiss Digital N/350D/REBEL XT.

Target Object

Target object	Access type	Data type number	Data type

Revision History/Date		Corrections	Reviser	Remarks

EdsImageRef	Read/Write	kEdsDataType_UInt32	EdsUInt32

Note: Write is available as the access type with EdsImageRef only for RAW images.

Value

Values defined in Enum EdsPhotoEffect.

Enum EdsPhotoEffect <defined location>EDSDKTypes.h

Value	Description
0	Off (Color Effect deactivated. Normal shooting.)
5	Black and white

See Also

- Regarding RAW support for each camera model, to determine if a property is valid during processing, see Support Status for RAW Properties.

Note

- With this property, it is possible to get values at the time of shooting.

5.2.47 kEdsPropID_FilterEffect

Description

Indicates the monochrome filter effect.
The supported models are the Kiss Digital N/350D/REBEL XT and 20D only.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_UInt32	EdsUInt32
EdsImageRef	Read		

Note: Write is available as the access type with EdsImageRef only for RAW images.

Value

Values defined in Enum EdsFilterEffect.

Enum EdsFilterEffect <defined location>EDSDKTypes.h

Value	Description
0	None
1	Yellow
2	Orange
3	Red
4	Green

See Also

- Regarding RAW support for each camera model, to determine if a property is valid during processing, see Support Status for RAW Properties.

Note

- This property is invalid for models supporting picture styles. For models supporting picture styles, use the property kEdsPropID_PictureStyleDesc.
- With this property, it is possible to get values at the time of shooting.

Revision History/Date		Corrections	Reviser	Remarks

5.2.48 kEdsPropID_ToningEffect

Description

Indicates the monochrome tone.
The supported models are the Kiss Digital N/350D/REBEL XT and 20D only.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_UInt32	EdsUInt32
EdsImageRef	Read/Write		

Note: Write is available as the access type with EdsImageRef only for RAW images.

Value

Value	Description
0	None
1	Sepia
2	Blue
3	Violet
4	Green
0xffffffff	Unknown

See Also

- Regarding RAW support for each camera model, to determine if a property is valid during processing, see Support Status for RAW Properties.

Note

- This property is invalid for models supporting picture styles. For models supporting picture styles, use the property kEdsPropID_PictureStyleDesc.
- With this property, it is possible to get values at the time of shooting.

5.2.49 kEdsPropID_ToneCurve

Description

Note: This property is unsupported. Do not use it.

Indicates the tone curve.
If the target object is EdsCameraRef, designate the following values in inParam.

Value of inParam	Description
0	Standard
1	Set 1
2	Set 2
3	Set 3

Note: If the target object is EdsImageRef, designate 0 in inParam.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_Uint32	EdsUInt32
EdsImageRef	Read		

Revision History/Date		Corrections	Reviser	Remarks

Value (EdsCameraRef)

Value	Description
0	Default
1	User-Defined 1
2	User-Defined 2
3	User-Defined 3

Value (EdsImage)

Value	Description
0x00000000	Standard
0x00000011	User setting
0x00000080	Custom setting
0x00000001	TCD1
0x00000002	TCD2
0x00000003	TCD3

5.2.50 kEdsPropID_PictureStyle

Description

Note: This property is unsupported. Do not use it.

Indicates the picture style.
This property is valid only for models supporting picture styles (the EOS 5D or EOS 1D Mark II N or later).
To get or set the picture style registered in "User Setting," designate user setting 1– (kEdsPictureStyle_User1–) in inParam. By using inParam = 0, you can designate the current picture style.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_UInt32	EdsUInt32
EdsImageRef	Read/Write		

Note: Write is available as the access type with EdsImageRef only for RAW images.

Value

Values defined in Enum EdsPictureStyle.
However, kEdsPictureStyle_UserX in Enum EdsPictureStyle is not used here.

Enum EdsPictureStyle <defined location>EDSDKTypes.h

Value	Picture style
0x0081	Standard
0x0082	Portrait
0x0083	Landscape
0x0084	Neutral
0x0085	Faithful
0x0086	Monochrome
0x0041	Computer Setting 1 (base picture style only)

Revision History/Date		Corrections	Reviser	Remarks

0x0042	Computer Setting 2 (base picture style only)
0x0043	Computer Setting 3 (base picture style only)

See Also

- Regarding RAW support for each camera model, to determine if a property is valid during processing, see Support Status for RAW Properties.

Note

- Computer settings (1 and so on) refers to data that was set by designating a picture style file to upload to the camera from a host computer. Computer setting data is registered in the corresponding user setting. (For example, computer setting 1 corresponds to user setting 1). As a user setting, it represents a picture style that users can select.
- Picture styles registered in computer settings always have a base picture style. As for picture styles other than presets, only base picture styles can be retrieved by means of this property value.
- With this property, it is possible to get values at the time of shooting.

5.2.51 kEdsPropID_PictureStyleDesc

Description

Indicates settings for each picture style.
This property is valid only for models supporting picture styles (the EOS 5D or EOS 1D Mark II N or later).

With **EdsGetPropertyData or EdsSetPropertyData**, you can designate a picture style in inParam to set that picture style setting item. By using inParam = 0, you can designate the current picture style.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_PictureStyleDesc	EdsPictureStyleDesc
EdsImageRef	Read/Write		

Note: Write is available as the access type with EdsImageRef only for RAW images.

Value

Value	Picture style
An integer from –4 to 4	Contrast
An integer from 0 to 7	Sharpness
An integer from –4 to 4	Saturation
An integer from –4 to 4	Color tone
0: None 1: Yellow 2: Orange 3: Red 4: Green 0xFFFFFFFF: Unknown	Monochrome filter effect
0: None 1: Sepia 2: Blue 3: Violet 4: Green 0xFFFFFFFF: Unknown	Monochrome tone

Revision History/Date		Corrections	Reviser	Remarks

See Also

- Regarding RAW support for each camera model, to determine if a property is valid during processing, see Support Status for RAW Properties.

Note

- Write is available as the access type with EdsImageRef objects only for RAW images. Processed images are read-only.
- With this property, it is possible to get values at the time of shooting.

5.2.52 kEdsPropID_UserWhiteBalanceData

Description

Note: This property is unsupported. Do not use it.

Indicates user-specified white balance data.
The supported models are EOS 1 series cameras only.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_ByteBlock	EdsInt8[]

Value

Data structures and related information is not disclosed.

5.2.53 kEdsPropID_UserToneCurveData

Description

Indicates user-specified tone curve data.
This property applies to EOS 1D Mark II and EOS 1Ds Mark II models.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_ByteBlock	EdsInt8

Value

Data structures and related information is not disclosed.
Allocate a suitable amount of memory for the client and store a suitable data set in EdsUsersetData data.

5.2.54 kEdsPropID_UserPictureStyleData

Description

Indicates user-specified picture style data in user settings (1 and so on).
User-specified picture styles refer to data that was set by designating a picture style file to upload to the camera from a host computer.
Applicable models support picture styles.
Designate user setting 1–3 (kEdsPictureStyle_User1–kEdsPictureStyle_User3) in inParam.

Target Object

Revision History/Date		Corrections	Reviser	Remarks

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_ByteBlock	EdsInt8[]

Value

Data structures and related information regarding user-specified picture style is not disclosed.
Allocate a suitable amount of memory for the client and store a suitable data set in EdsUsersetData data.

Note

- To set a user-specified picture style obtained from a picture style file on the camera, use EdsSetPropertyData to write to this property.
- In data retrieved with this property, you can get caption information (names) in user-specified picture style data.

5.2.55 kEdsPropID_FlashOn

Description

Indicates if the flash was on at the time of shooting.

Target Object

Target object	Access type	Data type number	Data type
EdsImageRef	Read	kEdsDataType_Uint32	EdsUInt32

Value

Value	Description
0	No flash
1	Flash

5.2.56 kEdsPropID_FlashMode

Description

Indicates the flash type at the time of shooting.

Target Object

Target object	Access type	Data type number	Data type
EdsImageRef	Read	kEdsDataType_Uint32_Array	EdsUInt32[]

Value

Array number	Description	Value
0	Flash type	0 = None (the "flash type" item itself is not displayed) 1 = Internal 2 = external E-TTL 3 = external A-TTL 0xFFFFFFFF = Invalid value
1	Synchro timing	0 = 1st Curtain Synchro 1 = 2nd Curtain Synchro 0xFFFFFFFF = Invalid value

Revision History/Date		Corrections	Reviser	Remarks

5.2.57 kEdsPropID_RedEye

Description

Indicates red-eye reduction.

Target Object

Target object	Access type	Data type number	Data type
EdsImageRef	Read	kEdsDataType_Uint32	EdsUInt32

Value

Value	Description
0	Off
1	On
0xFFFFFFFF	Invalid value

5.2.58 kEdsPropID_NoiseReduction

Description

Indicates noise reduction.

Target Object

Target object	Access type	Data type number	Data type
EdsImageRef	Read	kEdsDataType_Uint32	EdsUInt32

Value

Value	Description
0	Off
1	On 1
2	On 2
3	On
4	Auto

Note

• Values 1–3 vary depending on the camera model.

5.2.59 kEdsPropID_PictureStyleCaption

Description

Returns the user-specified picture style caption name at the time of shooting.
This property is valid only for models supporting picture styles (the EOS 5D or EOS 1D Mark II N or later).
User-specified picture styles refer to picture styles for which picture style files are read on a host computer and set on a camera.

Target Object

Target object	Access type	Data type number	Data type
EdsImageRef	Read	kEdsDataType_String	EdsChar[]

Revision History/Date		Corrections	Reviser	Remarks

Value

ASCII text strings up to 32 characters, including null-terminated strings.

5.2.60 kEdsPropID_SaveTo

Description

Indicates the destination of images after shooting.

Target Object

Target object	Access type	Data type number	Data type
EdsCameraRef	Read/Write	kEdsDataType_UInt32	EdsUInt32

Value

Values defined in Enum EdsSaveTo.

Enum EdsSaveTo <defined location>EDSDKTypes.h

Value	Description
1	Save on a memory card of a remote camera
2	Save by downloading to a host computer
3	Save both ways

Note

- If kEdsSaveTo_Host or kEdsSaveTo_Both is used, the camera caches the image data to be transferred until DownloadComplete or CancelDownload APIs are executed on the host computer (by an application). The application creates a callback function to receive camera events. If kEdsObjectEvent_DirItemRequestTransfer or kEdsObjectEvent_DirItemRequestTransferDT events are received, the application must execute DownloadComplete (after downloading) or CancelDownload (if images are not needed) for the camera.

5.2.61 kEdsPropID_LensName

Description

Returns the lens name at the time of shooting.
This property can only be retrieved from images shot using models supporting picture styles (the EOS 5D or EOS1D Mark II N or later).

Target Object

Target object	Access type	Data type number	Data type
EdsImageRef	Read	kEdsDataType_String	EdsChar[]

Value

Returns the lens name as an ASCII string.

Revision History/Date		Corrections	Reviser	Remarks

5.3 Support Status for RAW Properties

Support status by model is as follows regarding processing system properties of image objects.
Of the properties listed here, kEdsPropID_ClickWBPoint supports writing only. **As for other processing system properties, those indicated by ○ and ▲ can all be read or written.** ▲ indicates that it is possible to incorporate settings data in images in the stream (that is, to save the designated value in an image file) by means of **EdsReflectImageProperty**.

Property ID	Model							
	D30	D60	1D/1Ds	10D/Kiss D	1D Mark II/1Ds Mark II	20D/ Kiss Digital N/350D/REBEL XT	5D/30D/Kiss Digital X/400D/REBEL XTi	1D Mark II N
kEdsPropID_Linear	○	○	○	○	○	○	○	○
kEdsPropID_DigitalExposure	–	○	▲	○	▲	▲	▲	▲
kEdsPropID_WhiteBalance	○	○	▲	○	▲	▲	▲	▲
kEdsPropID_ColorTemperature	–	–	▲	○	▲	▲	▲	▲
kEdsPropID_WhiteBalanceShift	–	–	–	–	▲	▲	▲	▲
kEdsPropID_ClickWBPoint	○	○	○	○	○	○	○	○
kEdsPropID_WBCoeffs	–	–	○	–	○	–	–	○
kEdsPropID_ColorMatrix	–	–	▲	–	▲	–	–	–
kEdsPropID_Contrast	○	○	–	○	▲	▲	–	–
kEdsPropID_ColorSaturation	○	○	–	○	▲	▲	–	–
kEdsPropID_ColorTone	–	○	–	○	▲	▲	–	–
kEdsPropID_Sharpness	○	○	▲	○	▲	▲	–	–
kEdsPropID_ColorSpace	–	–	–	○	▲	▲	▲	▲
kEdsPropID_PhotoEffect	–	–	–	–	–	▲	–	–
kEdsPropID_FilterEffect	–	–	–	–	–	▲	–	–
kEdsPropID_ToningEffect	–	–	–	–	–	▲	–	–
kEdsPropID_PictureStyle	–	–	–	–	–	–	▲	▲
kEdsPropID_PictureStyleDesc	–	–	–	–	–	–	▲	▲

○ : Supported as a function
▲: Supported as a function, and the setting state can be recorded in a file
– : Not supported as a function

Revision History/Date		Corrections	Reviser	Remarks

6. Appendix

6.1 Using the EDSDK

In order to install an application built using EDSDK on a computer where it will be executed, that computer must be set up as an environment that can execute EDSDK for the application installer.

Windows version

Be sure to copy all EDSDK modules into the application sub folder.

Note1:

Be absolutely sure when you overwrite the old version of the library whenever a new version of EDSDK becomes available. We recommend that you copy files while comparing file versions of the library.

Note2:

Do not copy the EDSDK module to the Windows System folder or Windows folder.

Note3:

In order to connect to an EOS digital camera, the correct device driver software must be installed and a connection between the camera and the host PC must be established. (Driver software is not needed when using a camera model that performs PTP communications.) For details, see the installation method for drivers in the software installation guide included with your EOS digital camera.

Macintosh version

Be sure to copy EDSDK.framework into the application folder.

${AppFolder}/Contents/frameworks/

*Do not individually change or delete files in the EDSDK.framework folder.

Note1:

Be absolutely sure when you overwrite the old version of the library whenever a new version of EDSDK becomes available. We recommend that you copy files while comparing file versions of the library.

Note2:

Do not copy the EDSDK module to extention folders in addition to system folders.

Revision History/Date		Corrections	Reviser	Remarks

6.2 Data Types Used by the APIs

Data types defined under EDSDK are listed in EDSDKTypes.h in C language format.
This section introduces data types unique to EDSDK that are used by EDSDK APIs.

*For the most recent type definitions, see the header file EDSDKTypes.h.

6.2.1 EdsDirectoryItemInfo

This structure represents directory item information for the memory card in the camera.
It is specified as an argument to EdsGetDirectoryItemInfo.

```
typedef   struct   tagEdsDirectoryItemInfo {
            EdsUInt32 size;
            EdsBool                 isFolder;
            EdsUInt32 groupID;   // Type 2 protocol standard camera
            EdsUInt32 option;     // Type 2 protocol standard camera EdsTransferOption
            EdsChar                 szFileName[ EDS_MAX_NAME ];
} EdsDirectoryItemInfo;
```

6.2.2 EdsPropertyDesc

This structure represents a list of settable property data.
It is specified as an argument to EdsGetPropertyDesc.

```
typedef   struct   tagEdsPropertyDesc {
            EdsInt32    form;
            EdsAccess access;
            EdsInt32    numElements;
            EdsInt32    propDesc[128];
}EdsPropertyDesc;
```

6.2.3 EdsPoint

This structure is generally used to represent a set of coordinates.

```
typedef struct tagEdsPoint {
            EdsInt32      x;
            EdsInt32      y;
} EdsPoint;
```

6.2.4 EdsSize

This structure generally represents the width and height of a rectangle.

```
typedef   struct   tagEdsSize {
              EdsInt32        width;
              EdsInt32        height;
} EdsSize;
```

Revision History/Date		Corrections	Reviser	Remarks

6.2.5 EdsRect

This structure is generally used to indicate the coordinates of a rectangle.

```
typedef    struct    tagEdsRect {
                EdsPoint        point;
                EdsSize         size;
} EdsRect;
```

6.2.6 EdsImageInfo

This structure represents various information found in image data.
It is specified as an argument to EdsGetImageInfo.

```
typedef    struct    tagEdsImageInfo{
                EdsUInt32 width;                    // image width
                EdsUInt32 height;                   // image height

                EdsUInt32 numOfComponents; // number of color components in image.
                EdsUInt32 componentDepth;       // bits per sample.    8 or 16.

                EdsRect                 effectiveRect;          // Effective rectangles except
                                                                // a black line of the image.
                                                                // A black line might be in the top and bottom
                                                                // of the thumbnail image.
                EdsUInt32 reserved1;  // Reserved 1
                EdsUInt32 reserved2;  // Reserved 2
}EdsImageInfo;
```

6.2.7 EdsTime

This structure represents the camera time or the shooting date of an image.
It is used to store kEdsPropID_DateTime property data.

```
typedef    struct    tagEdsTime{
        EdsUInt32     year;                 // year
        EdsUInt32     month;                // month 1=January, 2=February, ...
        EdsUInt32     day;                  // day
        EdsUInt32     hour;                 // hour
        EdsUInt32     minute;                   // minute
        EdsUInt32     second;                   // second
        EdsUInt32     milliseconds;                     // reserved
} EdsTime;
```

6.2.8 EdsFocusPoint

This structure represents the AF frame information of focus information.
It stores AF frame information of the kEdsPropID_FocusInfo property.

```
    typedef    struct    tagEdsFocusPoint{
                EdsUInt32               valid;              // if the frame is valid.
```

Revision History/Date		Corrections	Reviser	Remarks

```
        EdsUInt32 justFocus;        // if the frame is just focus.
        EdsRect   rect;             // rectangle of the frame.
        EdsUInt32 reserved;         // reserved
} EdsFocusPoint;
```

6.2.9 EdsFocusInfo

This structure represents focus information.
It stores kEdsPropID_FocusInfo property data.

```
typedef  struct  tagEdsFocusInfo {
        EdsRect     imageRect;      // rectangle of the image.
        EdsUInt32   pointNumber;    // number of frames.
        EdsFocusPoint  focusPoint[128];  // each frame's description.
}EdsFocusInfo;
```

6.2.10 EdsRational

This structure is generally used to represent fractions.
It is used with many properties such as kEdsPropID_Av and kEdsPropID_Tv.

```
typedef  struct  tagEdsRational {
        EdsInt32  numerator;
        EdsUInt32 denominator;
} EdsRational;
```

6.2.11 EdsUsersetData

This structure is generally used to represent user-defined data.
This structure is used with properties such as kEdsPropID_UserPictureStyleData.

```
typedef  struct  tagEdsUsersetData {
        EdsUInt32 valid;
        EdsUInt32     dataSize;
        EdsChar       szCaption[32];
        EdsUInt8      data[1];
}EdsUsersetData;
```

6.2.12 EdsSaveImageSetting

Use this structure as an argument to EdsSaveImage.

```
typedef struct tagEdsSaveImageSetting{
        EdsUint32  JPEGQuality;     // 1 (coarse)～10 (fine)
        EdsStreamRef iccProfileStream;
        EdsUInt32 reserved ;
} EdsSaveImageSetting;
```

Revision History/Date		Corrections	Reviser	Remarks

6.2.13 EdsPictureStyleDesc

Use this structure when retrieving picture styles.

```
typedef struct tagEdsPictureStyleDesc {
        EdsInt32                contrast;
        EdsUInt32               sharpness;
        EdsInt32                saturation;
        EdsInt32                colorTone;
        EdsUInt32               filterEffect;
        EdsUInt32               oningEffect;
}EdsPictureStyleDesc;
```

Revision History/Date		Corrections	Reviser	Remarks

6.3 Sample Code

This sample code is written in C++.

6.3.1 SAMPLE1 From initializing to finalizing

```
void    applicationRun()
{
        EdsError err = EDS_ERR_OK;
        EdsCameraRef    camera = NULL;
        bool isSDKLoaded = false;

        // Initialize SDK
        err = EdsInitializeSDK();
        if(err = EDS_ERR_OK)
        {
                isSDKLoaded = true;
        }

        // Get first camera
        if(err = EDS_ERR_OK)
        {
                err = getFirstCamera (&camera);
        }

        // Set event handler
        if(err = EDS_ERR_OK)
        {
                err = EdsSetObjectEventHandler(camera,    kEdsObejctEvent_All,
                                                          handleObjectEvent,    NULL);
        }

        // Set event handler
        if(err = EDS_ERR_OK)
        {
                err = EdsSetPropertyEventHandler(camera,    kEdsPropertyEvent_All,
                                                          handlePropertyEvent,    NULL);

        }

        // Set event handler
        if(err = EDS_ERR_OK)
        {
                err = EdsSetPropertyEventHandler(camera,    kEdsStateEvent_All,
                                                          handleSateEvent,    NULL);
        }

        // Open session with camera
        if(err = EDS_ERR_OK)
        {
```

Revision History/Date		Corrections	Reviser	Remarks

```
                err = EdsOpenSession(camera);
        }

        /////
        // do something
        ////

        // Close session with camera
        if(err = EDS_ERR_OK)
        {
                err = EdsCloseSession(camera);
        }

        // Release camera
        if(camera != NULL)
        {
                EdsRelease(camera);
        }

        // Terminate SDK
        if(isSDKLoaded)
        {
                EdsTerminateSDK();
        }
}


EdsError EDSCALLBACK handleObjectEvent( EdsObjectEvent event,
                                        EdsBaseRef    object,
                                        EdsVoid * context)
{
        // do something

        /*
          switch(event)
        {
                case kEdsObjectEvent_DirItemRequestTransfer:
                        downloadImage(object);
                        break;

                default:

                        break;
        }
        */


        // Object must be released
        if(object)
        {
                EdsRelease(object);
        }
}
```

Revision History/Date		Corrections	Reviser	Remarks

```
EdsError EDSCALLBACK    handleSateEvent (EdsPropertyEvent event,
                                                 EdsPropertyID    property,
                                                 EdsVoid * context)
{
        // do something
}

EdsError EDSCALLBACK    handleSateEvent (EdsCameraStateEvent event,
                                                 EdsUInt32 parameter,
                                                 EdsVoid * context)
{
        // do something
}
```

6.3.2 SAMPLE2 Getting a camera object

```
EdsError    getFirstCamera(EdsCameraRef    *camera)
{
        EdsError err = EDS_ERR_OK;
        EdsCameraListRef    cameraList = NULL;
        EdsUInt32    count = 0;

        // Get camera list
        err = EdsGetCameraList(&cameraList);

        // Get number of cameras
        if(err == EDS_ERR_OK)
        {
                err = EdsGetChildCount(cameraList,    &count);
                if(count == 0)
                {
                        err = EDS_ERR_DEVICE_NOT_FOUND;
                }
        }
        // Get first camera retrieved
        if(err == EDS_ERR_OK)
        {
                err = EdsGetChildAtIndex(cameraList ,    0 ,    camera);
        }

        // Release camera list
        if(cameraList != NULL)
        {
                EdsRelease(cameraList);
                cameraList = NULL;
        }
}
```

Revision History/Date		Corrections	Reviser	Remarks

6.3.3 SAMPLE3 Getting a property

```
EdsError  getTv(EdsCameraRef  camera,  EdsUInt32  *Tv)
{
        EdsError err = EDS_ERR_OK;
        EdsUInt32  dataType;
        EdsUInt32  dataSize;

        err = EdsGetPropertySize(camera,  kEdsPropID_Tv,  0 ,  &dataType,  &dataSize);

        if(err == EDS_ERR_OK)
        {
                err = EdsGetPropertyData(camera,  kEdsPropID_Tv,  0 ,  dataSize,  Tv);
        }

        return err;
}
```

6.3.4 SAMPLE4 Getting a propertydesc

```
EdsError  getTvDesc(EdsCameraRef  camera,  const EdsPropertyDesc  *TvDesc)
{
        EdsError err = EDS_ERR_OK;

        err = EdsGetPropertyDesc(camera,  kEdsPropID_Tv,  TvDesc);

        return err;
}
```

6.3.5 SAMPLE5 Setting a property

```
EdsError  setTv(EdsCameraRef  camera,  EdsUInt32  TvValue)
{
        err = EdsSetPropertyData(camera,  kEdsPropID_Tv,  0 , sizeof(TvValue),  &TvValue);
}
```

6.3.6 SAMPLE6 Downloading an image

```
EdsError downloadImage(EdsDirectoryItemRef  directoryItem)
{
        EdsError err = EDS_ERR_OK;
        EdsStreamRef stream = NULL;

        // Get directory item information
        EdsDirectoryItemInfo  dirItemInfo;
```

Revision History/Date		Corrections	Reviser	Remarks

```
        err = EdsGetDirectoryItemInfo(directoryItem, & dirItemInfo);

          // Create file stream for transfer destination
        if(err == EDS_ERR_OK)
        {
                err = EdsCreateFileStream( dirItemInfo.szFileName,
                                                kEdsFile_CreateAlways,
                                                kEdsAccess_ReadWrite, &stream);
        }

        // Download image
        if(err == EDS_ERR_OK)
        {
                err = EdsDownload( directoryItem, dirItemInfo.Size, stream);
        }

        // Issue notification that download is complete
        if(err == EDS_ERR_OK)
        {
                err = EdsDownloadComplete(directoryItem);
        }

        // Release stream
        if( stream != NULL)
        {
                EdsRelease(stream);
                stream = NULL;
        }

        return err;
}
```

6.3.7 SAMPLE7 Getting a file object

```
EdsError  getVolume(EdsCameraRef  camera,  EdsVolumeRef *  volume)
{
        EdsError err = EDS_ERR_OK;
        EdsUInt32 count = 0;

        // Get the number of camera volumes
        err = EdsGetChildCount(camera,  &count);
                if(err == EDS_ERR_OK  && count == 0)
        {
                err =EDS_ERR_DIR_NOT_FOUND;
        }

        // Get initial volume
        if(err == EDS_ERR_OK)
        {
                err = EdsGetChildAtIndex(camera,  0 ,  &volume);
```

Revision History/Date		Corrections	Reviser	Remarks

```
        }
}
```

6.3.8 SAMPLE8 Getting DCIM Folder

```
EdsError getDCIMFolder(EdsVolumeRef    volume,    EdsDirectoryItemRef *    directoryItem)
{
        EdsError err = EDS_ERR_OK;
        EdsDirectoryItemRef    dirItem = NULL;
        EdsDirectoryItemInfo    dirItemInfo;
        EdsUInt32 count = 0;

        // Get number of items under the volume
        err = EdsGetChildCount(volume,    &count);
            if(err == EDS_ERR_OK    && count == 0)
        {
                err =EDS_ERR_DIR_NOT_FOUND;
        }

        // Get DCIM folder
        if(int i = 0 ; i < count && err == EDS_ERR_OK; i++)
        {
                // Get the ith item under the specifed volume
                if(err == EDS_ERR_OK)
                {
                        err = EdsGetChildAtIndex(volume,    i ,    &dirItem);
                }

                // Get retrieved item information
                if(err == EDS_ERR_OK)
                {
                        err = EdsGetDirectoryItemInfo(dirItem,    &dirItemInfo)
                }

                // Indicates whether or not the retrieved item is a DCIM folder.
                if(err == EDS_ERR_OK)
                {
                        if( stricmp(dirItemInfo.szFileName,    "DCIM") == 0 &&
                                dirItemInfo.isFolder == true)
                        {
                                directoryItem = dirItem;
                                break;
                        }
                }

                // Release retrieved item
                if(dirItem)
                {
                        EdsRelease(dirItem);
                        dirItem = NULL;
                }
        }

        return err;
}
```

Revision History/Date		Corrections	Reviser	Remarks

6.3.9 SAMPLE9 Taking a picture

```
EdsError takePicture(EdsCameraRef camera)
{
        return EdsSendCommand(kEdsCameraCommand_TakePicture , 0);
}
```

Revision History/Date		Corrections	Reviser	Remarks