INTERNATIONAL STANDARD

ISO 15740

Third edition
2013-09-15

Photography — Electronic still picture imaging — Picture transfer protocol (PTP) for digital still photography devices

Photographie — Imagerie des prises de vue électroniques — Protocole de transfert d'images (PTP) pour les appareils photographiques électroniques numériques

Reference number
ISO 15740:2013(E)

© ISO 2013

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

COPYRIGHT PROTECTED DOCUMENT

© ISO 2013

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Contents

Page

Foreword ..... **v**

Introduction ..... **vi**

1 Scope **1**

2 Normative references **1**

3 Terms and definitions **1**

4 Digital still photography device model **5**
- 4.1 Overview 5
- 4.2 Baseline requirements 6

5 Data format specification **6**
- 5.1 General format 6
- 5.2 Data types 7
- 5.3 Simple types 9
- 5.4 Arrays 11
- 5.5 Data sets 12

6 Image and data object formats **21**
- 6.1 Object usage 21
- 6.2 Thumbnail formats 22
- 6.3 ObjectFormatCodes 23
- 6.4 Object format version identification 23
- 6.5 Data object association 24

7 Transport requirements **26**
- 7.1 Disconnection events 26
- 7.2 Reliable, error-free channel 27
- 7.3 Asynchronous event support 27
- 7.4 Device discovery and enumeration 27
- 7.5 Specific transports 27

8 Persistent storage **27**
- 8.1 StorageID 27
- 8.2 Data object referencing 28
- 8.3 Receiver object placement 29

9 Communication protocol **30**
- 9.1 Device roles 30
- 9.2 Sessions 30
- 9.3 Transactions 30
- 9.4 Operation flow 33
- 9.5 Vendor extensions 33

10 Operations **35**
- 10.1 Operation overview 35
- 10.2 Operation parameters 35
- 10.3 OperationCode format 35
- 10.4 OperationCode summary 35
- 10.5 Operation descriptions 35

11 Responses **60**
- 11.1 ResponseCode format 60
- 11.2 ResponseCode summary 60
- 11.3 Response descriptions 61

12 Events **66**
- 12.1 Event usage 66
- 12.2 Event types 66

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

12.3 Event data set 66
12.4 EventCode format 67
12.5 EventCode summary 67
12.6 Event descriptions 67

13 Device properties 71
13.1 Device property usage 71
13.2 Values of a device property 71
13.3 Device property management requirements 72
13.4 Device property identification 72
13.5 Device property descriptions 76

14 Streaming (PTP v1.1 only) 92
14.1 Streaming overview 92
14.2 Stream transfer 92
14.3 Multiplexing 92
14.4 Discovering and configuring stream capabilities 93
14.5 Data transfer mechanism 93
14.6 Packet layout 94
14.7 Frame layout 95
14.8 Enumerating supported streams 95
14.9 Retrieving stream information 95

15 Conformance section 95

Annex A (informative) **Optional device features 98**

Annex B (normative) **Object referencing and format codes 100**

Annex C (informative) **Operation flow example scenarios 102**

Annex D (informative) **Filesystem implementation examples 106**

Annex E (informative) **Reference to OSI model 109**

Annex F (informative) **SendObject implementation example 112**

Bibliography 115

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2. www.iso.org/directives

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received. www.iso.org/patents

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

The committee responsible for this document is ISO/TC 42, *Photography*.

This third edition cancels and replaces the second edition (ISO 15740:2008), of which it constitutes a minor revision with the following changes:

— as the vendor extension ID registry formerly maintained by the I3A has been transferred to another organization, term 3.21 (I3A) was removed and the remaining terms renumbered;

— in 9.5.1, the fourth and fifth sentences were amended and combined to reflect that a new organization assigns and maintains VendorExtensionIDs.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Introduction

This third edition of ISO 15740 (hereinafter designated PTP v1.1) provides optional support for new increased performance and compatibility. All new constructs are fully backward compatible with the first edition (hereinafter designated PTP v1.0) and are optional. See 5.5.2 for standard version.

For the purposes of this International Standard, digital still photography devices (DSPDs) are defined as devices with persistent storage which capture a digital two-dimensional image at a discrete point in time. Most DSPDs include interfaces that can be used to connect to a host computer or other imaging device, such as a printer. A number of high speed interface transports has been developed, including USB, TCP/IP and IEEE 1394 (FireWire). This International Standard is designed to provide requirements for communicating with DSPDs. This includes communications with any type of device, including host computers, direct printers and other DSPDs over a suitable transport. The requirements include standard image referencing behaviour, operations, responses, events, device properties, data sets and data formats to ensure interoperability. This International Standard also provides optional operations and formats, as well as extension mechanisms.

This International Standard specifies the following:

— behaviour requirements for DSPDs; this includes the baseline features a device needs to support in order to provide interoperability over conforming transports;

— functional requirements needed by a transport to facilitate the creation of a transport-dependent implementation specification that conforms to this International Standard;

— a high-level protocol for communicating with and between DSPDs consisting of operation, data and response phases;

— sets of suggested data codes and their usages including

  — OperationCodes,

  — ResponseCodes,

  — ObjectFormatCodes,

  — DevicePropCodes,

  — EventCodes,

  — required data sets and their usages,

  — a means of describing data object associations and filesystems and

  — mechanisms for implementing extensibility.

This International Standard does not attempt to define any of the following:

— any sort of device discovery, enumeration or transport aggregation methods; implementation of this functionality is left to the transports and the platforms upon which support for this International Standard is implemented;

— an application programming interface; this is left to the platforms upon which support for this International Standard is implemented.

This International Standard has been designed to appropriately support popular image formats used in digital still cameras, including the Exif and TIFF/EP formats defined in ISO 12234-1[15] and ISO 12234-2, as well as the Design Rule for Camera Filesystem (DCF) and the Digital Print Order Format (DPOF).

The technical content of this International Standard is closely related to PIMA 15740:2000. The main difference is that PIMA 15740:2000 includes an informative annex describing a USB implementation of

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

ISO 15740. This information is not included in this International Standard, which instead references the USB still device class document developed by the Device Working Group of the USB Implementers Forum.

PTP v1.1 provides optional support for new increased performance and compatibility. All new constructs are fully backward compatible with PTP v1.0 and are optional.

— Performance Enhancements:

  — Support for retrieval of ObjectHandles in enumerated chunks, via specification of three new optional operations and a new response code. This may reduce long response times for some initiators that possess large numbers of objects.

  — Support for optional arbitrary resizing prior to image transmission via specification of a new operation GetResizedImageObject. In PTP v1.0, image sizes might be requested in full-resolution or thumbnail size only.

  — Support for arrays of data sets. This can be used to reduce the number of required transactions necessary for device characterization from being a function of the number of objects on the device to one.

  — An optional fast file characterization operation called GetFilesystemManifest that exploits data set arrays to request, in a single transaction, only the minimum data required to characterize a typical filesystem. Many initiators, particularly in printing scenarios, are interested in fast filesystem characterization for access to a specifically named file in a particular place. This capability can significantly improve end-user workflow latency. This single operation replaces the typical series of many GetObjectInfo requests with a binary filesystem manifest. This manifest is defined as a simple array of a subset of the standard ObjectInfo data set called the ObjectFilesystemInfo data set. This operation replaces the need for many GetObjectInfo calls, while also avoiding the need for responders to perform many internal file-opens on the fly, or to cache ObjectInfo image data that is often held persistently only “inside” internal image files (e.g. TIFF tags inside EXIF JPEGs), to quickly communicate only the fast filesystem information.

— Compatibility Enhancements:

  — An optional mechanism to support multiple vendor extension sets. This is specified via the new VendorExtensionMap data set, and two new optional operations that may be invoked outside of a session (GetVendorExtensionMaps and GetVendorDeviceInfo).

  — The optional fast file characterization method GetFilesystemManifest natively supports extremely large objects, by requiring 8-bytes for object size (UINT64), as opposed to the standard 4-bytes.

  — A new standard ObjectFormatCode to support the Digital Negative file format (DNG).

— Feature Enhancement:

  — An optional mechanism for handling streaming content. This is specified via the new StreamInfo data set, as well as the supporting GetStreamInfo and GetStream operations, as well as some optional new supporting DeviceProperties. This is described in a new Clause 14.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

--`,,``````,,```,,,```,````,`,-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Photography — Electronic still picture imaging — Picture transfer protocol (PTP) for digital still photography devices

1 Scope

This International Standard provides a common communication protocol for exchanging images with and between digital still photography devices (DSPDs). This includes communication between DSPDs and host computers, printers, other digital still devices, telecommunications kiosks and image storage and display devices.

This protocol is transport- and platform-independent.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 8601, *Data elements and interchange formats — Information interchange — Representation of dates and times*

ISO 12234-2, *Electronic still-picture imaging — Removable memory — Part 2: TIFF/EP image data format*

ISO/IEC 10646, *Information technology — Universal Coded Character Set (UCS)*

ISO/IEC 10918-1:1994, *Information technology — Digital compression and coding of continuous-tone still images: Requirements and guidelines*

IEC 61966-2-1, *Multimedia systems and equipment — Colour measurement and management — Part 2-1: Colour management — Default RGB colour space — sRGB*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1
album
end-user-created object used to logically group data objects according to some user-defined criteria

Note 1 to entry: An album might or might not be a physical folder in a filesystem. In this International Standard, an album is a type of association.

3.2
association
logical construct used to expose a relationship between discrete objects

Note 1 to entry: Associations are used to indicate that separate data objects are related. Associations are represented like folders, and can be nested using a standard branched hierarchical tree structure.

EXAMPLE A time sequence, or user-defined groupings by content or capture session.

3.3
connection
transport-provided mechanism for establishing paths for transferring data between devices

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

3.4
datacode
16-bit unsigned integer whose Most Significant Nibble (4 bits) is used to indicate the category of code and whether the code value is standard or vendor-extended

3.5
data object
image or other type of data that typically exists in persistent storage of a DSPD or other device

3.6
dataset
transport-independent collection of one or more individual data items with known interpretations

Note 1 to entry: Data sets are not necessarily opaque nor atomic to transport implementations.

3.7
Design Rule for Camera Filesystem
DCF
standard convention for camera filesystems which specifies the file format, foldering and naming conventions in order to promote file interoperability between conforming digital photography devices

3.8
device discovery
act of determining the set of all devices present on a particular transport or platform that are physically or logically accessible

3.9
digital still photography device
DSPD
device with persistent storage which captures a two-dimensional digital still image

3.10
Digital Print Order Format
DPOF
standardized ASCII file stored on removable media along with the image files that indicates how many copies of which images should be printed

Note 1 to entry: DPOF also allows index prints, cropping, and text overlays to be specified.

3.11
enumeration
act of creating an ordered increasing numerical list that contains one representative element for each member of a set

3.12
Exif/JPEG
compressed file format for digital cameras in which the images are compressed using the baseline JPEG standard described in ISO 12234-2

Note 1 to entry: In Exif, metadata and thumbnail images are stored using TIFF tags within an application segment at the beginning of the JPEG file.

3.13
folder
optional sub-structure in a hierarchical storage area that can contain data objects

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

3.14
FlashPix
image file format, defined in *FlashPix Format Specification*, using a structured storage file containing metadata and a tiled, hierarchical image representation

Note 1 to entry: The tiles in a FlashPix image are normally baseline JPEG images, and individual image tiles of a particular resolution can be easily accessed for rapid display and editing.

3.15
IEEE 1394
high-speed serial bus standardized by the IEEE (Institute of Electrical and Electronics Engineers) currently having clock rates of 100, 200 and 400 Mbits/s

Note 1 to entry: IEEE 1394 is often referred to as FireWire.

3.16
image aspect ratio
ratio of the image width to the image height

3.17
image capture device
device for converting a scene or a fixed image, such as a print, film or transparency, to digital image data

3.18
image output device
device that can render a digital image to hardcopy or softcopy media

3.19
in-band event
event transmitted on the same logical connection as operations and responses

Note 1 to entry: Events are only asynchronous to the degree of data precision for which the transport implementation allows event interleaving.

3.20
initiator
device that initiates a conversation by opening a session, and issues all formal operations to the responder

Note 1 to entry: The initiator is analogous to the client in the client/server paradigm.

3.21
Infrared Data Association
IrDA
infrared wireless communication system that currently supports wireless communication at data rates between 9 600 bps and 4 Mbps

3.22
Joint Photographic Experts Group
JPEG
specific image compression method defined in ISO/IEC 10918-1

3.23
LogicalStorageID
least significant sixteen bits of a StorageID

Note 1 to entry: This value uniquely identifies one logical storage area within the physical store indicated in the PhysicalStorageID.

3.24
Most Significant Nibble
MSN
most significant four bits of the most significant byte

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

3.25
object aggregation
act of taking one or more location-specific lists of objects that exist on a particular device and grouping them together in one set

3.26
ObjectHandle
device-unique 32-bit unsigned integer assigned by a device to each data object in local persistent storage which is provided to external devices

Note 1 to entry: External recipients of an ObjectHandle must use it to reference that piece of data in subsequent transactions. ObjectHandles are guaranteed to be persistent over at least a session.

3.27
out-of-band event
event transmitted on a different logical connection to that for operations and responses

Note 1 to entry: Out-of-band events are asynchronous from operation transactions.

3.28
personal computer
PC
any personal computing device, which may employ various hardware architectures and operating systems

3.29
PhysicalStorageID
most significant sixteen bits of a StorageID

Note 1 to entry: This value uniquely identifies one physical storage area on a device, although there may be more than one logical store per physical store.

3.30
Portable Network Graphics
PNG
extensible file format for lossless, portable, compressed storage of raster images

Note 1 to entry: PNG supports indexed colour, greyscale, truecolour and an optional alpha channel.

3.31
protocol
defined mechanisms for exchanging data between devices

3.32
pull model
use paradigm for DSPDs where the object receiver initiates the operation requests to transfer data objects from the sender

3.33
push model
use paradigm for DSPDs where the object sender initiates the operation requests to transfer data objects to the receiver

3.34
QuickDraw picture
file format consisting of sequences of saved drawing commands

Note 1 to entry: QuickDraw files are commonly referred to as PICT files.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

3.35
responder
device that responds to operations from the initiator

Note 1 to entry: The responder is analogous to a server in the client/server paradigm.

3.36
session
logical connection between two devices defining a period of time during which obtained state information, such as handle persistence, may be relied upon

3.37
square pixel sampling
image having equal sample spacing in the two orthogonal sampling directions

3.38
StorageID
device-specific four-byte unsigned integer (UINT32) that represents a unique storage area that may contain data objects

Note 1 to entry: The most significant 16 bits of a StorageID represent the PhysicalStorageID, while the least significant 16 bits of a StorageID represent the LogicalStorageID.

3.39
transport aggregation
act of taking one or more transport-specific list of conforming devices that are logically or physically accessible in a system and grouping them in one set that spans all transports across the particular system

3.40
transport
means of attaching the digital capture device to some other digital device including a physical wire or a wireless connection

3.41
Universal Serial Bus
USB
digital interface for connecting up to 127 devices in a tiered-star topology

4 Digital still photography device model

4.1 Overview

Digital still photography devices (DSPDs) are used to acquire digitally encoded still images. These devices include a persistent storage capability so that any digital images and other data acquired by the device are preserved across power cycle operations unless they are specifically deleted.

A DSPD might support many different features. This International Standard supports devices with a wide range of potential features. However, a small number of features is required for conformance with this International Standard, while many others are optional. Subclause 4.2 describes the required features and functionality. Annex A describes features that are not required for conformance but which should be implementable using this International Standard and its extension mechanisms.

Standard data formats for datatypes and data sets are described in Clause 5.

Clause 6 describes required and optional support for particular image and non-image formats and metadata. Clause 6 also describes methods for associating data objects.

A particular feature set places requirements on the transports used to connect the DSPD to other devices. Clause 7 describes these requirements.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

All DSPDs must store images in some form of storage area. Clause 8 describes the usage of these stores, as well as the methods for referencing them.

Clause 9 describes the roles of devices, sessions and transactions that transports are required to use in order to communicate with and/or between DSPDs. Clause 10 lists the standard operations, their corresponding optional operation codes and their usages. Standard responses to operations are defined in Clause 11. The use of events is mandatory in order to ensure synchronization between devices. Clause 12 describes events and their usages.

In order to expose device controls and manipulate properties in a common way, a standard set of device properties and their usages have been defined in Clause 13.

Clause 15 serves as a summary of the individual operations and events that are required to be supported by particular devices, as well as a checklist that can be used by implementers.

4.2 Baseline requirements

4.2.1 General

The requirements listed in 4.2.2 to 4.2.5 shall be met in order for a DSPD to conform to this International Standard.

4.2.2 Implementation of a suitable transport

The DSPD shall provide appropriate hardware and software support for at least one transport that meets the requirements specified in Clause 7.

4.2.3 Thumbnail support

The DSPD shall provide support for thumbnails as described in 6.2.

4.2.4 Standard image and data reference behaviour

In order to ensure interoperability, it is necessary to define a standard mechanism for describing image and data objects present on a device. The DSPD shall meet the requirements described in Clause 6.

4.2.5 Asynchronous event support

The DSPD shall be capable of generating and reacting to asynchronous events. Clause 12 describes events and their usages.

5 Data format specification

5.1 General format

5.1.1 Multibyte data

For the purposes of interpretability, all data fields showing internal content representations shall be read from left to right, in order of decreasing byte significance, commonly referred to as big-endian notation. Therefore, the left-most byte shall represent the Most Significant Byte (MSB), and the right-most byte shall represent the Least Significant Byte (LSB). The most significant four bits of the MSB are referred to as the Most Significant Nibble (MSN), while the least significant four bits of the LSB are referred to as the Least Significant Nibble (LSN). The actual multibyte format used on the wire is transport-specific, while the actual multibyte format used at the application interface is platform-specific.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

5.1.2 Bit format

Bit fields presented in this International Standard are numbered so that the least significant bit is at the zero position, holding the right-most position in the field; e.g. the most significant bit of a UINT32 would be referred to as bit 31, while the least significant bit would be referred to as bit 0.

5.1.3 Hexadecimal notation

This International Standard uses hexadecimal notation as a means of concisely describing multibyte fields. All hexadecimal byte fields are represented with the prefix "0x". Following this prefix are pairs of characters, where each pair represents one byte, with the most significant byte appearing first and the least significant byte appearing last.

5.2 Data types

5.2.1 Datatype summary

The types of data that are defined in this International Standard as having specific interpretations of their data content are listed in Table 1.

Table 1 — Datatype summary

Name	Size (bytes)	Format	PTP Version
OperationCode	2	Datacode (UINT16)	1.0+
ResponseCode	2	Datacode (UINT16)	1.0+
EventCode	2	Datacode (UINT16)	1.0+
DevicePropCode	2	Datacode (UINT16)	1.0+
ObjectFormatCode	2	Datacode (UINT16)	1.0+
StorageID	4	Special (UINT32)	1.0+
ObjectHandle	4	Handle (UINT32)	1.0+
DateTime	Variable	String	1.0+
DeviceInfo	Variable	Dataset	1.0+
StorageInfo	Variable	Dataset	1.0+
ObjectInfo	Variable	Dataset	1.0+
DevicePropDesc	Variable	Dataset	1.0+
DevicePropDescEnum	Variable	Enumerated form of DevicePropDesc	1.0+
DevicePropDescRange	Variable	Range form of DevicePropDesc	1.0+
Object	Variable	Variable	1.0+
VendorExtensionMap	8	Dataset	1.1+
ObjectFilesystemInfo	Variable	Dataset	1.1+
StreamInfo	36	Dataset	1.1+
EnumID	4	Special (UINT32)	1.1+

5.2.2 Datacodes

Datacodes are 16-bit unsigned integers (UINT16) with specified interpretations, used for the purposes of enumeration. In order to aid in visual interpretation and potential transport debugging, and to simplify some transport implementations, the primary and vendor-defined datacodes for operations, responses,

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

data formats, events, and properties in this International Standard have mutually exclusive values. The most significant four bits of a datacode (Most Significant Nibble) shall have a particular bit pattern that identifies its code type. Therefore, the allocation of these four bits to type specification infers that the minimum value of any enumerated datacode is 0 (xxxx0000-00000000) and the maximum value is 4,095 (xxxx1111-11111111).

It is strongly recommended that transport implementations use these codes directly in their binary representations, but this is not mandatory. Particular transport implementations may be unable to use the specified code systems for one or more code types, due to pre-existing structure formats for data-wrapping, or other constraints. Where it is possible to use the codes, they should be used. If one or more particular datacode types cannot be used, the transport implementation specification should still attempt to accommodate those datacode types that can be used. If the binary form suggested in this International Standard is not used for a particular datacode type, an appropriate corresponding enumerated identifier in an alternate form should be made available where possible for each datatype enumeration specified, each having the same usage and definition as those specified in this International Standard. This allows for transport-aggregating abstractions in host software to use the codes defined in this International Standard, even though a particular code might not be transmitted across the wire for a particular transport in the binary form specified. Transports may also need to perform multiple transactions over the wire in order to fulfill one operation defined in this International Standard, and therefore one operation code may not be sufficient.

For example, if a transport does not use the 16-bit OperationCodes, it should still provide an equivalent mechanism for the GetObject operation that supports the same usage defined in this International Standard. Another example is a transport that uses OperationCodes for some operations but not others, because the transport in question possesses a built-in mechanism for performing the equivalent operation, and provides its own operation identification scheme for that operation. See Table 2.

Table 2 — Datacode formats

Bit 15	Bit 14	Bit 13	Bit 12	Bits 11-0	Code type
0	0	0	0	Any	Undefined (not a conforming code)
0	0	0	1	Any	Standard OperationCode
0	0	1	0	Any	Standard ResponseCode
0	0	1	1	Any	Standard ObjectFormatCode
0	1	0	0	Any	Standard EventCode
0	1	0	1	Any	Standard DevicePropCode
0	1	1	0	Any	Reserved
0	1	1	1	Any	Reserved
1	0	0	0	Any	Undefined
1	0	0	1	Any	Vendor-Defined OperationCode
1	0	1	0	Any	Vendor-Defined ResponseCode
1	0	1	1	Any	Vendor-Defined ObjectFormatCode
1	1	0	0	Any	Vendor-Defined EventCode
1	1	0	1	Any	Vendor-Defined DevicePropCode
1	1	1	0	Any	Reserved
1	1	1	1	Any	Reserved

It is a convention of this International Standard that all datacodes shall set bit 15 to 1 in order to indicate that the code value is vendor-specific, and therefore undefined in this International Standard. Codes indicating that they are vendor-defined should be interpreted according to the VendorExtensionID and VendorExtensionVersion fields of the DeviceInfo data set as described in 5.5.2.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Individual datacode interpretations and usage are described in the appropriate section of this International Standard for each type of datacode.

5.3 Simple types

5.3.1 Simple type summary

The generic datatypes that may be used in this International Standard are listed in Table 3.

All datatypes having bit 14 set to 1 are uniform arrays of individual fixed-length types.

Table 3 — Datatype codes

Datatype code	Type	Description
0x0000	UNDEF	Undefined
0x0001	INT8	Signed 8-bit integer
0x0002	UINT8	Unsigned 8-bit integer
0x0003	INT16	Signed 16-bit integer
0x0004	UINT16	Unsigned 16-bit integer
0x0005	INT32	Signed 32-bit integer
0x0006	UINT32	Unsigned 32-bit integer
0x0007	INT64	Signed 64-bit integer
0x0008	UINT64	Unsigned 64-bit integer
0x0009	INT128	Signed 128-bit integer
0x000A	UINT128	Unsigned 128-bit integer
0x4001	AINT8	Array of signed 8-bit integers
0x4002	AUINT8	Array of unsigned 8-bit integers
0x4003	AINT16	Array of signed 16-bit integers
0x4004	AUINT16	Array of unsigned 16-bit integers
0x4005	AINT32	Array of signed 32-bit integers
0x4006	AUINT32	Array of unsigned 32-bit integers
0x4007	AINT64	Array of signed 64-bit integers
0x4008	AUINT64	Array of unsigned 64-bit integers
0x4009	AINT128	Array of signed 128-bit integers
0x400A	AUINT128	Array of unsigned 128-bit integers
0xFFFF	STR	Variable-length unicode string
All other values with bit 15 set to 0	Undefined	Reserved
All other values with bit 15 set to 1	Vendor-defined	Vendor-defined

5.3.2 Integers

The most common data type that is required in this International Standard is an integer. Integers may be signed or unsigned, and may be placed into arrays.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

5.3.3 Handles

Handles are 32-bit device-unique unsigned integers (UINT32) that are exposed externally in order to allow consistent referencing to its logical and/or physical elements by other conforming devices. All handles act as UINT32 datatypes with the added constraint that they have individually unique values relative to the currently open session. Handles shall be persistent over at least a particular session, and may or may not be persistent over an entire power cycle, or even across power cycles. There is no significance to the value of a handle other than that it is unique, and therefore they do not need to be consecutively assigned. The values 0x00000000 and 0xFFFFFFFF may not refer to any valid objects, and are reserved for context-specific meanings, such as "no handle," "default handle" or "all handles." Any use of context-specific meanings shall be described in the appropriate clauses.

Handles are used to refer to image and non-image objects that are present on a device as well as objects that are expected to immediately become present, and in this context are referred to as ObjectHandles. ObjectHandles may refer to image or non-image objects that are capable of being retrieved as the response to an operation. The existence of an ObjectHandle infers the ability to produce an ObjectInfo data set for the object that is referred to, as well as production of the object itself for object types that are not fully qualified by the ObjectInfo data set. An association (e.g. a generic folder) is an example of an object type that is fully qualified by an ObjectInfo data set, and therefore possesses no actual object to send. The type of data object that the ObjectHandle refers to may be determined by examining the ObjectFormatCode described in the data object's ObjectInfo data set described in 5.5.3.

5.3.4 Decimal types

Fixed-point datatypes may be represented using integers by contextually specifying a scalar factor. This protocol does not currently require conventions for handling floating-point data. Transmitted data of this type are typically contained inside a data object that is opaque to this protocol.

5.3.5 Strings

5.3.5.1 String format

All strings shall consist of standard 2-byte unicode characters (USC-2) encoded with UTF-16 as described in ISO/IEC 10646, with the added constraint of having a maximum number of characters of 255. Strings shall be represented by a combination of two fields as described in Table 4. An unused string field shall be represented by specifying a length field of zero (0x00) and including no actual string bytes, resulting in a total PTP-encoded string length of 1 byte.

Table 4 — String format

Dataset field	Size (bytes)	Datatype
NumChars	1	UINT8
StringChars	Variable	Unicode null-terminated string

NumChars: Represents the number of characters in the string, including the terminating null character for non-empty strings. Permitted values are 0 to 255.

StringChars: This field holds the actual unicode null-terminated string with 2-byte characters. This field shall not contain more than 255 characters, including the terminating null characters. This field is not present for empty strings.

5.3.5.2 DateTime String

When needed, the date and time shall be expressed using a compatible subset of ISO 8601 so that it can be easily parseable. This shall take the form of a unicode string in the format "YYYYMMDDThhmmss.s" where YYYY is the year, MM is the month 01–12, DD is the day of the month 01–31, T is a constant character, hh is the hours since midnight 00–23, mm is the minutes, 00–59, past the hour, and ss.s is the

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

seconds past the minute, with the ".s" being optional tenths of a second past the second. This string can optionally be appended with Z to indicate UTC, or ±hhmm to indicate that the time is relative to a time zone. Appending neither indicates that the time zone is unspecified.

5.4 Arrays

5.4.1 Simple arrays

A simple array is a concatenation of one or more elements of the same fixed-length type. All array data sets contain a field representing the total number of elements contained in the array. See Table 5. The datatype, element size and interpretation of the individual elements shall be context-specific, and therefore not explicit within the array structure. An empty array would consist of only a single UINT32 value of 0x00000000. For internal array referencing purposes, arrays shall be treated as zero-based.

Table 5 — Array format

Field	Size (bytes)	Format
NumElements	4	UINT32
ArrayEntry[0]	ElementSize	Special
ArrayEntry[1]	ElementSize	Special
...	ElementSize	...
ArrayEntry[NumElements-1]	ElementSize	Special

NumElements: Represents the total number of elements contained in the array, and therefore is equal to the number of ArrayEntry fields. This value may be 0x00000000, which indicates an empty array that would only be four total bytes in size.

ArrayEntry (*n*): The total number of these fields shall equal the value held by the NumElements field. The size and interpretation (i.e. datatype) of each ArrayEntry is context-dependent, but should be the same for all elements in an array. These fields are not present for empty arrays.

5.4.2 Data set arrays (PTP v1.1)

PTP v1.1 provides a standard for encoding arrays of data sets, and relies upon their usage for some optional performance and compatibility enhancements, such as those provided by the GetFilesystemManifest and GetVendorExtensionMaps operations.

Arrays of data sets are encoded in a manner similar to fixed-length types described in 5.4.1, having a "header" field that holds the number of objects in the array. A single data set array may hold any number of objects of a single data set type, within the constraints of 8 bytes. Arrays for data sets, being new to PTP v1.1, are slightly different in that they use 64 bits (i.e. UINT64 instead of UINT32) to hold the number of objects. As with the PTP v1.0 standard, the datatype, element size and interpretation of the individual elements of the data sets in the array shall be context-specific, and therefore not explicit within the array structure. See Table 6.

If the individual data sets in the array correspond to particular objects on the device, each of the data sets will require a field to hold an ObjectHandle, in order to distinguish which data set describes which object.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table 6 — Data set array format

Field	Size (bytes)	Format
NumElements	8	UINT64
ArrayEntry[0]	Fixed or Variable	Dataset
ArrayEntry[1]	Fixed or Variable	Dataset
...	Fixed or Variable	...
ArrayEntry[NumElements-1]	Fixed or Variable	Dataset

5.5 Data sets

5.5.1 Data set usage

For the purposes of this International Standard, the term data set is defined as a transport-independent collection of one or more individual data items with defined interpretations. A data set serves as a set of requirements for the data that must be accounted for by an opposing device as the result of a particular operation or event. Individual data items may hold other embedded data items. A transport implementation may or may not need to place its own wrappers around the data set, split data sets into multiple pieces, pack the fields a particular way or pass different fields using different mechanisms.

In order to avoid inconsistencies when defining their implementations of the data sets, transport implementations should reference the field ordering in this specification. If the transport is wrapping the data set completely, it should retain the field ordering defined in this specification unless a distinct benefit can be obtained by not doing so. The particular byte packing mechanism for individual fields must be clearly defined, such as little-endian or big-endian.

All unused fields in all data sets shall initially be set to their default value. If a field does not have an explicit default value, the implicit default value is zero for integer datatypes, and the empty string for strings. Any fields in a data set which are unused are considered to have no default value, and should be set to zero.

If data sets are changed in future versions of this International Standard, fields will only be added, never deleted. No existing fields will be redefined in future versions of this International Standard, and the specified field ordering shall remain constant. Unless otherwise indicated, reserved fields in data sets are required to be transmitted by the transport implementation, but should be set to 0x00000000 for integers and to an empty string for strings.

5.5.2 DeviceInfo data set

See Table 7.

This data set is used to hold the description information for a device. The initiator can obtain this data set from the responder without opening a session with the device. This data set holds data that describe the device and its capabilities. This information is only static if the device capabilities cannot change during a session, which would be indicated by a change in the FunctionalMode value in the data set; e.g. if the device goes into a sleep mode in which it can still respond to GetDeviceInfo requests, the data in this data set should reflect the capabilities of the device while it is in that mode only (including any operations and properties needed to change the FunctionalMode, if this is allowed remotely). If the power state or the capabilities of the device change (due to a FunctionalMode change), a DeviceInfoChanged event shall be issued to all sessions in order to indicate how its capabilities have changed.

All optional strings that are not provided should consist of an empty string. Subclause 5.3.5 describes the use of strings.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table 7 — DeviceInfo data set

Dataset field	Field order	Size (bytes)	Data type
StandardVersion	1	2	UINT16
VendorExtensionID	2	4	UINT32
VendorExtensionVersion	3	2	UINT16
VendorExtensionDesc	4	Variable	String
FunctionalMode	5	2	UINT16
OperationsSupported	6	Variable	OperationCode Array
EventsSupported	7	Variable	EventCode Array
DevicePropertiesSupported	8	Variable	DevicePropCode Array
CaptureFormats	9	Variable	ObjectFormatCode Array
ImageFormats	10	Variable	ObjectFormatCode Array
Manufacturer	11	Variable	String
Model	12	Variable	String
DeviceVersion	13	Variable	String
SerialNumber	14	Variable	String

StandardVersion: highest version of the standard that the device can support. This represents the standard version expressed in hundredths (e.g. 1.32 would be stored as 132). The version value used for PTP v1.0 is 1.00 (i.e. 100), whereas the version used for PTP v1.1 is 1.10 (i.e. 110).

NOTE The use of 1.00 (i.e. 100) for PTP v1.0 was not formally specified in PTP v1.0, and was assumed (i.e. neither the original PIMA 15740:2000 nor ISO 15740:2005 specified the value of this field). As part of the standardization of PTP v1.1, all PTP initiators should interpret any value for Standard Version less than 110 (0, 1, 2, ..., 108, 109) as PTP v1.0, and anything larger than 110 as undefined.

VendorExtensionID: provides the context for interpretation of the default set of vendor extensions used by this device. These IDs are assigned by PIMA, as described in 9.5. If no extensions are supported, this field shall be set to 0x00000000. If vendor-specific codes of any type are used, this field is mandatory, and should not be set to 0x00000000. If exactly one set of vendor extensions is supported, the device would not support the PTP v1.1 GetVendorExtensionMaps or GetVendorDeviceInfo operations, and this value would be interpreted as previously defined in PTP v1.0 (i.e. as the “one and only” “VendorExtensionID”). If multiple vendor extension sets are supported, those operations would both be supported, and this field is interpreted as the default, or “native” set, with all other supported sets being mapped into unused portions of the native set, as communicated to the initiator by the two new optional operations.

VendorExtensionVersion: the vendor-specific version number of extensions that are supported. This shall be expressed in hundredths (e.g. 1.32 would be stored as 132).

VendorExtensionDesc: an optional string used to hold a human-readable description of the VendorExtensionID. This field should only be used for informational purposes, and not as the context for the interpretation of vendor extensions.

FunctionalMode: an optional field used to hold the functional mode. This field controls whether the device is in an alternate mode that provides a different set of capabilities (i.e. supported operations, events, etc.). If the device only supports one mode, this value should always be zero. Table 8 describes the standard functional modes.

The functional mode information is held by the device as a device property. This property is described in 13.5.2. In order to change the functional mode of the device remotely, a session needs to be opened with the device, and the SetDeviceProp operation needs to be used.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table 8 — FunctionalMode values

Value	Description
0x0000	Standard mode
0x0001	Sleep state
All other values with bit 15 set to zero	Reserved
All values with bit 15 set to 1	Vendor-defined

OperationsSupported: This field is an array of OperationCodes representing operations that the device is currently supporting, given the FunctionalMode indicated. Subclause 10.3 describes these codes.

EventsSupported: This field is an array of EventCodes representing the events that are currently generated by the device in appropriate situations, given the FunctionalMode indicated. Subclause 12.6 describes these codes.

DevicePropertiesSupported: This field is an array of DevicePropCodes representing DeviceProperties that are currently exposed for reading and/or modification, given the FunctionalMode indicated. Subclause 13.5 describes these codes.

CaptureFormats: the list of data formats in ObjectFormatCode form that the device can create using an InitiateCapture operation and/or an InitiateOpenCapture operation, given the FunctionalMode indicated. These are typically image object formats, but can include any object format that can be fully captured using a single trigger mechanism, or an initiate/terminate mechanism. All image object formats in which a device can capture data shall be listed prior to any non-image object formats and shall be in preferential order such that the default capture format is first. ObjectFormats are described in Clause 6.

ImageFormats: the list of image formats in ObjectFormatCode form that the device supports in order of highest preference to lowest preference. Support for an image format refers to the ability to interpret image file contents, according to that format's specifications, for display and/or manipulation purposes. For image output devices, this field represents the image formats that the output device is capable of outputting. This field does not describe any device format translation capabilities. Refer to Clause 6 for more information on image format support.

Manufacturer: an optional human-readable string used to hold the responder's manufacturer.

Model: an optional human-readable string used to communicate the responder's model name.

DeviceVersion: an optional string used to communicate the responder's firmware or software version in a vendor-specific way.

SerialNumber: an optional string used to communicate the responder's serial number, which is defined as a unique value among all devices sharing identical Model and Device Version fields. If unique serial numbers are not supported, this field shall be set to the empty string. The presence of a non-null string in the SerialNumber field for one device infers that this field is non-zero and unique among all devices of that model and version.

5.5.3 ObjectInfo data set

This data set is used to define the information about data objects in persistent storage, as well as optional information if the data are known to be an image or an association object. It is required that these data items be accounted for in response to a GetObjectInfo operation. If the data are not known to be an image, or the image information is unavailable, the image-specific fields shall be set to zero. Objects of type Association are fully qualified by the ObjectInfo data set. See Table 9.

--``,,`````,,```,,,```,````,`,,`-`-`,,`,,`,`,,`---

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table 9 — ObjectInfo data set

Dataset field	Field order	Size (bytes)	Data type	Image specific	Association specific
StorageID	1	4	StorageID	No	No
ObjectFormat	2	2	ObjectFormatCode	No	No
ProtectionStatus	3	2	UINT16	No	No
ObjectCompressedSize	4	4	UINT32	No	No
ThumbFormat	5	2	ObjectFormatCode	Yes	No
ThumbCompressedSize	6	4	UINT32	Yes	No
ThumbPixWidth	7	4	UINT32	Yes	No
ThumbPixHeight	8	4	UINT32	Yes	No
ImagePixWidth	9	4	UINT32	Yes	No
ImagePixHeight	10	4	UINT32	Yes	No
ImageBitDepth	11	4	UINT32	Yes	No
ParentObject	12	4	ObjectHandle	No	No
AssociationType	13	2	AssociationCode	No	Yes
AssociationDesc	14	4	AssociationDesc	No	Yes
SequenceNumber	15	4	UINT32	No	No
Filename	16	Variable	String	No	No
CaptureDate	17	Variable	DateTime String	No	No
ModificationDate	18	Variable	DateTime String	No	No
Keywords	19	Variable	String	No	No

StorageID: the StorageID of the device's store in which the image resides. See 8.1 for a description of StorageIDs.

ObjectFormat: indicates ObjectFormatCode of the object. See 6.3 for a list of these codes.

ObjectCompressedSize: the size of the buffer, in bytes, needed to hold the entire binary object. This field may be used by transport implementations for memory allocation purposes in object receivers.

ProtectionStatus: an optional field representing the write-protection status of the data object. Objects that are protected may not be deleted as the result of any operations specified in this International Standard without first separately removing their protection status in a separate transaction. The values are enumerated according to Table 10.

Table 10 — ObjectInfo ProtectionStatus values

Value	Description
0x0000	No protection
0x0001	Read-only
All other values	Reserved

All values not explicitly defined are reserved for future use. This protection field is distinctly different in scope from the AccessCapability field present in the StorageInfo data set described in 5.5.4. If an attempt is made to delete an object, success will only occur if the ProtectionStatus of the object is 0x0000 and the AccessCapability of the store allows deletion. If a device does not support object protection, this field should always be set to 0x0000, and the SetProtection operation should not be supported. Refer to 5.5.4 for a description of the StorageInfo data set.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

ThumbFormat: indicates ObjectFormat of the thumbnail. In order for an object to be referred to as an image, it must be able to produce a thumbnail as the response to a request. Therefore, this value should only be 0x00000000 for the case of non-image objects. Refer to 6.3 for a list of ObjectFormatCodes.

ThumbCompressedSize: the size of the buffer needed to hold the thumbnail. This field may be used for memory allocation purposes. In order for an object to be referred to as an image, it must be able to produce a thumbnail as the response to a request. Therefore, this value should only be 0x00000000 for the case of non-image objects.

ThumbPixWidth: an optional field representing the width of the thumbnail in pixels. If this field is not supported or the object is not an image, the value 0x00000000 shall be used.

ThumbPixHeight: an optional field representing the height of the thumbnail in pixels. If this field is not supported or the object is not an image, the value 0x00000000 shall be used.

ImgPixWidth: an optional field representing the width of the image in pixels. If the data are not known to be an image, this field should be set to 0x00000000. The purpose of this field is to enable an application to provide the width information to a user prior to transferring the image. If this field is not supported, the value 0x00000000 shall be used.

ImgPixHeight: an optional field representing the height of the image in pixels. If the data are not known to be an image, this field should be set to 0x00000000. The purpose of this field is to enable an application to provide the height information to a user prior to transferring the image. If this field is not supported, the value 0x00000000 shall be used.

ImgBitDepth: an optional field representing the total number of bits per pixel of the uncompressed image. If the data are not known to be an image, this field should be set to 0x00000000. The purpose of this field is to enable an application to provide the bit depth information to a user prior to transferring the image. This field does not attempt to specify the number of bits assigned to particular colour channels, but instead represents the total number of bits used to describe one pixel. If this field is not supported, the value 0x00000000 shall be used. This field should not be used for memory allocation purposes, but is strictly information that is typically inside an image object that may affect whether or not a user wishes to transfer the image, and is therefore exposed prior to object transfer in the ObjectInfo data set.

ParentObject: indicates the handle of the object that is the parent of this object. The ParentObject must be of object type Association. If the device does not support associations, or the object is in the “root” of the hierarchical store, then this value should be set to 0x00000000.

AssociationType: a field that is only used for objects of type Association. This code indicates the type of association. Refer to 6.5 for a description of associations and a list of defined types. If the object is not an association, this field should be set to 0x0000.

AssociationDesc: This field is used to hold a descriptor parameter for the association, and may therefore only be non-zero if the AssociationType is non-zero. The interpretation of this field is dependent upon the particular AssociationType, and is only used for certain types of association. If unused, this field should be set to 0x00000000. Refer to 6.5 for information on this descriptor.

SequenceNumber: This field is optional, and is only used if the object is a member of an association, and only if the association is ordered. If the object is not a member of an ordered association, this value should be set to 0x00000000. These numbers should be created consecutively. However, to be a valid sequence, they do not need to be consecutive, but only monotonically increasing. Therefore, if a data object in the sequence is deleted, the SequenceNumbers of the other objects in the ordered association do not need to be renumbered, and examination of the sequential numbers will indicate a possibly deleted object by the missing sequence number.

Filename: an optional string representing filename information. This field should not include any filesystem path information, but only the name of the file or directory itself. The interpretation of this string is dependent upon the FilenameFormat field in the StorageInfo data set that describes the logical storage area in which this object is stored. See 5.5.4 for information on this field.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

CaptureDate: a static optional field representing the time that the data object was initially captured. This is not necessarily the same as any date held in the ModificationDate field. This data set uses the DateTime string described in 5.3.5.2.

ModificationDate: an optional field representing the time of last modification of the data object. This is not necessarily the same as the CaptureDate field. This data set uses the DateTime string described in 5.3.5.2.

Keywords: an optional string representing keywords associated with the image. Each keyword shall be separated by a space. A keyword that consists of more than one word shall use underscore (_) characters to separate individual words within one keyword.

5.5.4 StorageInfo data set

This data set is used to hold the state information for a storage device. See Table 11.

Table 11 — StorageInfo data set

Dataset field	Field order	Length (bytes)	Data type
StorageType	1	2	UINT16
FilesystemType	2	2	UINT16
AccessCapability	3	2	UINT16
MaxCapacity	4	8	UINT64
FreeSpaceInBytes	5	8	UINT64
FreeSpaceInImages	6	4	UINT32
StorageDescription	7	Variable	String
VolumeLabel	8	Variable	String

StorageType: the code that identifies the type of storage, particularly whether the store is inherently random-access or read-only memory, and whether it is fixed or removable media. See Table 12.

All undefined values are reserved for future use.

Table 12 — Storage types

Code value	Storage type
0x0000	Undefined
0x0001	Fixed ROM
0x0002	Removable ROM
0x0003	Fixed RAM
0x0004	Removable RAM
All other values	Reserved

FilesystemType: This optional code indicates the type of filesystem present on the device. This field may be used to determine the filenaming convention used by the storage device, as well as to determine whether support for a hierarchical system is present. See Table 13. If the storage device is DCF-conformant, it shall be so indicated here.

All values having bit 31 set to zero are reserved for future use. If a proprietary implementation wishes to extend the interpretation of this field, bit 31 should be set to 1.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table 13 — FilesystemType values

Value	Description
0x0000	Undefined
0x0001	Generic flat
0x0002	Generic hierarchical
0x0003	DCF
All other values with bit 15 set to 0	Reserved
All values with bit 15 set to 1	Vendor-defined

AccessCapability: This field indicates whether the store is read-write or read-only. If the store is read-only, deletion may or may not be allowed. The permitted values are described in Table 14. Read-write is only valid if the StorageType is non-ROM, as described in the StorageType field in Table 12.

All values having bit 15 set to zero are reserved for future use. If a proprietary implementation wishes to extend the interpretation of this field, bit 15 should be set to 1.

Table 14 — StorageInfo AccessCapability values

Value	Description
0x0000	Read-write
0x0001	Read-only without object deletion
0x0002	Read-only with object deletion
All other values	Reserved

MaxCapacity: This is an optional field that indicates the total storage capacity of the store in bytes. If this field is unused, it should report 0xFFFFFFFF.

FreeSpaceInBytes: the amount of free space that is available in the store in bytes. If this value is not useful for the device, it may set this field to 0xFFFFFFFF and rely upon the FreeSpaceInImages field instead.

FreeSpaceInImages: the number of images that may still be captured in this store according to the current image capture settings of the device. If the device does not implement this capability, this field should be set to 0xFFFFFFFF. This field may be used for devices that do not report FreeSpaceInBytes, or the two fields may be used in combination.

StorageDescription: an optional field that may be used for a human-readable text description of the storage device. This should be used for storage-type-specific information as opposed to volume-specific information. Examples would be “Type I Compact Flash,” or “3.5-inch 1.44 MB Floppy”. If unused, this field should be set to the empty string.

VolumeLabel: an optional field that may be used to hold the volume label of the storage device, if such a label exists and is known. If unused, this field should be set to the empty string.

5.5.5 VendorExtensionMap data set (PTP v1.1)

This data set is an optional data set defined in PTP v1.1, that is used to support multiple vendor extensions. This data set contains the mapping information for a single supported 16-bit (i.e. UINT16) vendor extended datacode of any of the standard types (Operation, Response, Event, DeviceProperty and ObjectFormat Codes). The mutual exclusion of all code category types means that the type of category may be determined from inspecting the value of the code as described in Table 2. All native and mapped code values should have the MSB (Most Significant Bit) set to 1 to indicate vendor extended status in the typical fashion of all vendor extended codes in this International Standard. See Table 15.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

These data sets are typically communicated in a data set array, as the response to the GetVendorExtensionMaps operation, which returns an entire mapping of one or more additional sets or subsets of vendor extensions, for devices that support multiple vendor extension sets, as described by Table 24. When used in this fashion, the "version" of the vendor extension set that is used for interpretation for a particular code is specified in the DeviceInfo data set retrieved for that particular extension set (i.e. using the new optional GetVendorDeviceInfo operation in 10.5.33).

Table 15 — VendorExtensionMap data set (PTP v1.1)

Dataset field	Field order	Length (bytes)	Data type
NativeCode	1	2	UINT16
MappedCode	2	2	UINT16
MappedVendorExtensionID	3	4	UINT32

NativeCode: The datacode in the "unused" portion of the "default" VendorExtension space, as determined by the responder. This is a single UINT16 Operation, Response, Event, ObjectFormat, or DeviceProp datacode with the vendor extended most significant bit 15 set to one, indicating its vendor-extended status. The specific type of datacode in the NativeCode field must match the type of datacode in the following MappedCode field of the same VendorExtensionMap data set instance.

MappedCode: The datacode of the non-default VendorExtension code in the "native" value with respect to its VendorExtensionID, as defined in that vendor extension specification. This is a single UINT16 Operation, Response, Event, ObjectFormat or DeviceProp datacode with the vendor extended most-signficant bit 15 set to one, indicating its vendor extended status. The specific type of datacode in the MappedCode field must match the type of datacode in the proceeding NativeCode field of the same VendorExtensionMap data set instance.

MappedVendorExtensionID: The VendorExtensionID (defined for the default set of vendor extensions and for PTP v1.0 devices by the UINT32 field in the standard DeviceInfo data set) for which this particular datacode is in context.

5.5.6 ObjectFilesystemInfo data set (PTP v1.1)

See Table 16.

This optional data set is a subset of the standard ObjectInfo data set, containing only the fields required to minimally characterize most basic filesystems. This data set is typically retrieved in an array using the optional GetFilesystemManifest operation to obtain, in a single transaction, all of the filesystem information available from a responder, while avoiding any image-specific metadata that may require time to acquire. Objects that support this optional operation must still support the standard GetObjectInfo operation, which still can and will be called subsequently for more information on particular objects at the initiator's discretion, as well as PTP v1.0 or performance insensitive initiators.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table 16 — ObjectFilesystemInfo data set (PTP v1.1)

Dataset field	Field order	Size (bytes)	Datatype	Assoc. specific	Equivalent ObjectInfo dataset field
ObjectHandle	1	4	ObjectHandle	No	N/A
StorageID	2	4	StorageID	No	1
ObjectFormat	3	2	ObjectFormatCode	No	2
ProtectionStatus	4	2	UINT16	No	3
ObjectCompressedSize64	5	8	UINT64	No	4[a]
ParentObject	6	4	ObjectHandle	No	12
AssociationType	7	2	AssociationCode	Yes	13
AssociationDesc	8	4	AssociationDesc	Yes	14
SequenceNumber	9	4	UINT32	No	15
Filename	10	Variable	String	No	16
ModificationDate	11	Variable	DateTime String	No	18

[a]The ObjectCompressedSize64 field is equivalent to the standard ObjectCompressedSize field in the ObjectInfo dataset, except that it uses a 64-bit integer in order to support objects of potentially very large size.

5.5.7 StreamInfo data set (PTP v1.1)

See Table 17.

Table 17 — StreamInfo data set (PTP v1.1)

Dataset field	Field order	Size (bytes)	Datatype
DatasetSize	1	8	UINT32
TimeResolution	2	8	UINT32
FrameHeaderSize	3	4	UINT32
FrameMaxSize	4	4	UINT32
PacketHeaderSize	5	4	UINT32
PacketMaxSize	6	4	UINT32
PacketAlignment	7	4	UINT32

TimeResolution: Contains the value in nanoseconds of one unit of the time stamp in the frame header. Thus, to calculate the time interval between two frames the following formula is used:

$$\text{ElapsedTime} = (\text{TimeStamp2} - \text{TimeStamp1}) \times \text{TimeResolution [ns]}$$

FrameHeaderSize: Contains the size, in bytes, of the frame header. The initiator should read the frame payload at this offset. For this specification, the size of the header is 8 bytes. This can be changed in future versions; therefore, this header will point to the payload, thus enabling the existing clients to work with the new protocol.

FrameMaxSize: The maximum size in bytes of the frame the initiator can expect including the header. The responder should not send frames larger than this. This is not the same as the entire length of the video stream, but is a format-specific construct abstracted for use in PTP. This will always be a fixed number.

PacketHeaderSize: Contains the size in bytes of the packet header. The initiator should read the packet payload at this offset. This is provided in order to enable future extensions to this protocol and make sure that the existing clients will work properly.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

PacketMaxSize: The maximum size in bytes of the packet the initiator can expect including the header. The responder should not send packets larger than this. If the frame is built up of multiple packets then all packets excluding the last should be of the same PacketMaxSize.

PacketAlignment: The number of bytes to which the packet should be aligned. The alignment may be required by some transport or device responder implementations in order achieve optimal performance.

6 Image and data object formats

6.1 Object usage

A data object is defined as an image or other type of data that exists in persistent storage of a DSPD or other device. This International Standard specifies how image and data objects are transferred between a digital still photography device (DSPD) and other devices, but it does not specify how they are stored within such devices. A DSPD conforming to this International Standard shall transfer images to other devices using a particular data or datafile format for storage. The DSPD shall identify the data format used for the main (e.g. full size) image transferred, and shall provide a thumbnail (e.g. reduced size) image using one of the two thumbnail image file formats defined in this clause. The image file formats used to transfer the main image shall be either

— Exif/JPEG version 2.2, the preferred format,

— one of the permitted image file formats listed in Table 18 or

— a proprietary image file format.

All devices conforming to this International Standard which receive images, shall be capable of receiving (but not necessarily displaying) image files using any of these file formats. In addition, if the receiving device displays images, it shall be capable of decoding and displaying both of the thumbnail image file formats defined in this clause.

For all image and data objects, an ObjectFormatCode is provided in the ObjectInfo data set to specify the format, as described in 5.5.3. The setting of bit 31 of this code can be used in conjunction with the VendorExtensionID in the DeviceInfo data set to create a vendor extended ObjectFormatCode to handle the transfer of non-standard file formats.

Table 18 — ObjectFormatCodes

Object Format-Code	Type	Format	Description
0x3000	A	Undefined	Undefined non-image object
0x3001	A	Association	Association (e.g. folder)
0x3002	A	Script	Device-model-specific script
0x3003	A	Executable	Device-model-specific binary executable
0x3004	A	Text	Text file
0x3005	A	HTML	HyperText Markup Language file (text)
0x3006	A	DPOF	Digital Print Order Format file (text)
0x3007	A	AIFF	Audio clip
0x3008	A	WAV	Audio clip
0x3009	A	MP3	Audio clip
0x300A	A	AVI	Video clip
0x300B	A	MPEG	Video clip
Type: Image File Format (I); Ancillary Data File Format (A)			

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table 18 *(continued)*

Object Format-Code	Type	Format	Description
0x300C	A	ASF	Microsoft Advanced Streaming Format (video)
0x300D	A	QuickTime	Apple QuickTime Video Format
0x300E	A	XML	eXtensible Markup Language file
0x3800	I	Undefined	Unknown image object
0x3801	I	Exif/JPEG	Exchangeable File Format, JEITA standard
0x3802	I	TIFF/EP	Tag Image File Format for Electronic Photography
0x3803	I	FlashPix	Structured Storage Image Format
0x3804	I	BMP	Microsoft Windows Bitmap file
0x3805	I	CIFF	Canon Camera Image File Format
0x3806	I	Undefined	Reserved
0x3807	I	GIF	Graphics Interchange Format
0x3808	I	JFIF	JPEG File Interchange Format
0x3809	I	PCD	PhotoCD Image Pac
0x380A	I	PICT	Quickdraw Image Format
0x380B	I	PNG	Portable Network Graphics
0x380C	I	Undefined	Reserved
0x380D	I	TIFF	Tag Image File Format
0x380E	I	TIFF/IT	Tag Image File Format for Image Technology
0x380F	I	JP2	JPEG2000 Baseline File Format
0x3810	I	JPX	JPEG2000 Extended File Format
0x3811	I	DNG	Digital Negative Format (PTP v1.1)
All other codes with MSN of 0011	Any	Undefined	Reserved for future use
All other codes with MSN of 1011	Any	Vendor-defined	Vendor-defined
Type: Image File Format (I); Ancillary Data File Format (A)			

6.2 Thumbnail formats

6.2.1 Permitted thumbnail formats

The thumbnail images provided by a DSPD shall be either compressed JPEG images or uncompressed TIFF images. Compressed JPEG images are preferred.

The device may optionally be capable of producing a thumbnail for non-still-image object formats, such as video clips, audio formats, etc. The method that a device should use to generate a thumbnail from these object types is not specified. This capability would be apparent if the device correctly responded to a GetThumb operation request for a data object with a DataFormat that is a non-still-image type.

6.2.2 Compressed JPEG thumbnail image files

The format of compressed thumbnail files shall be a JPEG interchange file using the baseline process specified in Table 1 of ISO/IEC 10918-1:1994. This requires that the image file use the following:

— DCT-based image compression process;

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

— 8-bit samples within each component;
— a baseline sequential (non-progressive) process;
— Huffman coding with 2 AC and 2 DC tables.

The baseline image file format is further restricted in that it shall have the following:

— square pixel sampling;
— a single interleaved scan (e.g. Y, Cr, Nb interleaved blocks);
— sRGB colour space as defined in IEC 61966-2-1;
— 3 colour components, Y, Cr, Nb, derived from the sRGB signals as follows, as specified in ITU-R BT.601[1)]:
  — Y = 0,299 R + 0,587 G + 0,114 B,
  — Nb = (− 0,299 R − 0,587 G + 0,886 B) × 0,564 + offset,
  — Cr = (0,701 R − 0,587 G − 0,114 B) × 0,713 + offset,
  — or a single (luminance) component;
— either 4:2:2 or 4:2:0 Y:Nb:Cr spatially centred colour subsampling or Y:Nb:Cr co-sited subsampling. Writers may use either option. Readers shall support both options.

NOTE These features are all requirements of the Exif version 2.2 specification.

In addition, it is preferred that the thumbnails meet the thumbnail image data format requirements specified in section 3.3.6 of Design Rule for Camera File system (DCF), version 2.0. These requirements include the following:

— JPEG compression using 4:2:2 chrominance sampling and the “typical” Huffman table provided in ISO/IEC 10918-1:1994;
— 160 × 120 pixel image record with a 4:3 aspect ratio.

6.2.3 Uncompressed TIFF thumbnail image files

The format of the uncompressed thumbnail images shall meet the requirements for the TIFF/EP format described in ISO 12234-2. The thumbnail image shall have a compression tag value = 1 (no compression) with the thumbnail image contained within IFD0.

6.3 ObjectFormatCodes

A list of object formats is given in Table 18. An ObjectFormatCode is a 16-bit unsigned integer (UINT16) that represents the format of an image or data object that resides on a DSPD. Setting bit 15 to 1 in the ObjectFormatCode indicates a vendor-defined format. All ObjectFormatCodes that represent image formats shall also have bit 11 set to 1, including those that are vendor-defined.

6.4 Object format version identification

The version of the object format should be contained in the data object in a context-specific manner. Support for a particular ObjectFormatCode implies that the device possesses the capability to read the internal structure of the data object in a way that is particular to its format, and this includes the discovery and interpretation of any version information that may exist. The format for specifying the version information is dependent upon the versioning methodology used by the particular format. Formats that change over time should remain backward compatible with older formats. However, if backward compatibility is not retained for a certain format, the specific format for that version and

1) Digital Video Standard (formerly CCIR 601).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

above should be considered as a separate format, and therefore a new ObjectFormatCode should be assigned to the new versions of that format.

6.5 Data object association

6.5.1 Association usage

This International Standard provides an optional method for associating related image and data objects, and this mechanism is what is used for representing folders and filesystems. Associations are represented using objects that are of type Association. All of the objects that are part of an association shall be in the branch of the object tree underneath the corresponding association, in a folder-like manner. The association to which an object belongs may be determined by examining the ParentObject fields of each object's ObjectInfo data set.

6.5.2 Association sub-types

There are different sub-types of associations. The sub-type is specified in the AssociationType field of the ObjectInfo data set of the association object. Table 19 lists the various associations that may be used.

Table 19 — Association sub-types

AssociationCode	AssociationType	AssociationDesc interpretation
0x0000	Undefined	Undefined
0x0001	GenericFolder	Unused
0x0002	Album	Reserved
0x0003	TimeSequence	DefaultPlaybackDelta
0x0004	HorizontalPanoramic	Unused
0x0005	VerticalPanoramic	Unused
0x0006	2DPanoramic	ImagesPerRow
0x0007	AncillaryData	Undefined
All other values with bit 15 set to 0	Reserved	Undefined
All values with bit 15 set to 1	Vendor-defined	Vendor-defined

GenericFolder: This association sub-type is used to represent a folder that may hold any type of object, and is analogous to the standard folder present in most filesystems. This association is typically used to represent a local grouping of objects, with no other relationship implied.

Album: This association sub-type is the same as a folder but is used to hold image and data objects that have logical groupings according to content, capture sessions or any other unspecified user-determined grouping. These are typically created by a user or automation technique. Some devices may wish to expose albums to the user but not all generic folders. Devices that do not distinguish between albums and folders should only use the AssociationType of GenericFolder. The AssociationDesc field is reserved for future definition by this International Standard and shall be set to 0x00000000 for devices that support this version of the specification.

TimeSequence: indicates that the data objects are part of a sequence of data captures that make up a set of time-ordered data of the same subject. This association is used to represent time-lapse or burst sequences. The order is interpreted to be sequential by capture time from first captured to last, and is indicated by the increasing values of the SequenceNumber fields in the ObjectInfo data set for each object. If known, the AssociationDesc acts as a DefaultPlaybackDelta, and should be set to the desired time, in milliseconds, in order to delay between each object if exposing them sequentially in real time. If unknown, this value should be set to 0x00000000.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

HorizontalPanoramic: indicates that the associated data objects make up a panoramic series of images that are arranged side by side, in a horizontal fashion. The order of the sequence, from left to right when facing the subject, is indicated by the increasing values of the SequenceNumber fields in each object. The AssociationDesc is unused, and should be set to 0x00000000; e.g. four images would have SequenceNumbers assigned as shown in Figure 1.

1	2	3	4

Figure 1 — HorizontalPanoramic SequenceNumber example

VerticalPanoramic: indicates that the associated data objects make up a panoramic series of images that are arranged bottom-to-top, in a vertical fashion. The order of the sequence, from bottom to top when facing the subject, is indicated by the increasing values of the SequenceNumber fields in each object. The AssociationDesc is unused, and should be set to 0x00000000; e.g. four images would have SequenceNumbers assigned as shown in Figure 2.

4
3
2
1

Figure 2 — VerticalPanoramic SequenceNumber example

2DPanoramic: indicates that the associated data objects make up a two-dimensional panoramic series of images that are arranged left-to-right and bottom-to-top in adjacent or overlapping horizontal strips. The order of the sequence, from bottom left to top right when facing the subject, is indicated by the increasing values of the SequenceNumber fields in each object. The AssociationDesc is used to indicate the number of images in each row; e.g. Sixteen images arranged in a 4 × 4 2DPanoramic would have SequenceNumbers assigned as shown in Figure 3.

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

Figure 3 — 2DPanoramic SequenceNumber example

AncillaryData: indicate that the association represents one or more non-image objects being associated with an image object; e.g. an image capture that also stores independent audio text files that are temporally related to the image capture may use this type of association to indicate the relationship. Optionally, if the individual objects are ordered (e.g. multiple temporally-related sound files), the SequenceNumber field of each object's ObjectInfo data set should contain increasing integers. If the individual objects are unordered, the SequenceNumbers for those objects should be set to 0x00000000. In this case, the AssociationDesc fields for those objects are unused, and should be set to 0x00000000.

--`,,`````,,```,,,```,````,`,-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

6.5.3 Unordered associations

Unordered associations are the standard type of association used to relate data objects non-sequentially, such as filesystem folders, albums, etc. Objects that belong to unordered associations always possess ObjectInfo data sets with SequenceNumber fields equal to zero.

6.5.4 Ordered associations

Ordered associations are an optional mechanism that may be used to represent associated objects, some or all of which are sequential in some context-specific fashion. Objects that are part of the ordered association may or may not have a non-zero SequenceNumber.

a) Objects of any type may be children of ordered associations, including other ordered or unordered associations.

b) Sequence numbers are used to order a proper or improper subset of child objects of a particular ordered association.

c) The value of a sequence number is only relevant between objects at the same level of the same hierarchical tree branch (i.e. siblings with the common ordered association parent).

d) The value of a sequence number is only persistent while the child–parent relationship is intact. If an object that is part of an ordered association is assigned a new parent, the object's SequenceNumber should be invalidated (i.e. set to zero or to a new appropriate SequenceNumber if the new parent is an ordered association).

e) Not all child objects of an ordered association need to have a sequence number. Objects without sequence numbers whose parent is an ordered association are considered related to the entire set of sequential objects, but are not a proper part of the sequence.

f) Non-zero sequence numbers among objects in an ordered association shall be unique.

6.5.5 Associations as filesystem folders

Unordered associations may be used to represent filesystem layouts of data objects within a store. This allows representation of filesystems that are not dependent upon particular pathname conventions. Each object contains a ParentObject field in its ObjectInfo data set. This ParentObject is the ObjectHandle of the association object that "contains" this data object. This mechanism serves to expose a bottom-up singly-linked list that may be used to reconstruct the filesystem hierarchy, which may be stored on the opposing device optionally using a different data structure (e.g. double-linked list, top-down enumerator, etc.). Only objects of type Association may serve as ParentObjects. The bottom-up nature of the child–parent link information results in ParentObjects that are stateless with regard to which objects are their children, and infers that if a child is deleted, the parent and the ObjectInfo data set describing it do not need to be updated. If the object exists in the "root" of the store, the value 0x00000000 shall be used for its ParentObject. Pathnames may be recreated using the Filename and ParentObject fields of each ObjectInfo data set, inserting the separators of choice, in a platform-specific manner. Some receiving devices may not support exposing a hierarchical structure, in which case objects of type Association would be ignored. In other cases, devices may only wish to show associations of type album as hierarchical. For an example of representing a filesystem using associations, see D.2.

7 Transport requirements

7.1 Disconnection events

The transport shall be capable of notifying the overlying agents or system software that a device has been disconnected from its previous environment. This environment may be either a wire interconnect or a wireless medium.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

7.2 Reliable, error-free channel

The transport shall provide a reliable connection between devices comprising a session, assuming that the connection is not physically broken or terminated. This may be provided in a transport-specific manner using error correction codes, retry scenarios and fault recovery.

7.3 Asynchronous event support

Since a conforming device is required to notify other devices about any changes in its operational status, configuration, available objects or stores, the transport shall provide a mechanism that supports the generation of outgoing events and servicing of incoming events asynchronously from operations, responses or data transfers. In order to generate events, some transports may require initiation of an operation while others may require a notification mechanism.

7.4 Device discovery and enumeration

Transports shall possess the capability of discovering and enumerating connected devices. Any transport may support a v1.0 or v1.1 PTP implementation.

7.5 Specific transports

7.5.1 USB

Devices using this transport shall conform to the following specifications:

— *USB Specification*, Version 1.1, USB Implementer's Forum, 23 September, 1998[2];

— *Universal Serial Bus Still Image Capture Device Definition*, Version 1.0, USB Implementer's Forum, 11 July, 2000[3].

7.5.2 IEEE 1394

Devices using the IEEE 1394 transport shall conform to the following specifications:

— IEEE 1394-1995[4], *IEEE Standard for High Performance Serial Bus*;

7.5.3 TCP/IP

Devices using the transport TCP/IP shall conform to the following specifications:

— *Transmission Control Protocol*, Internet Engineering Task Force (IETF) RFC 793[13], by Information Sciences Institute, University of Southern California, September 1981; http://ietf.org/rfc/rfc793.txt;

— *PTP-IP IP Picture Transfer Protocol*, Version 1.0, Camera and Imaging Products Association (CIPA) [14], Japan, Copyright 2005 by FotoNation Inc.

8 Persistent storage

8.1 StorageID

Every local persistent storage area on a device is represented by a unique, 4-byte, unsigned integer (UINT32) referred to as a StorageID. Each StorageID has two parts. The 16 most significant bits represent a physical storage device, while the 16 least significant bits represent a logical storage area within a physical store. See Figure 4.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

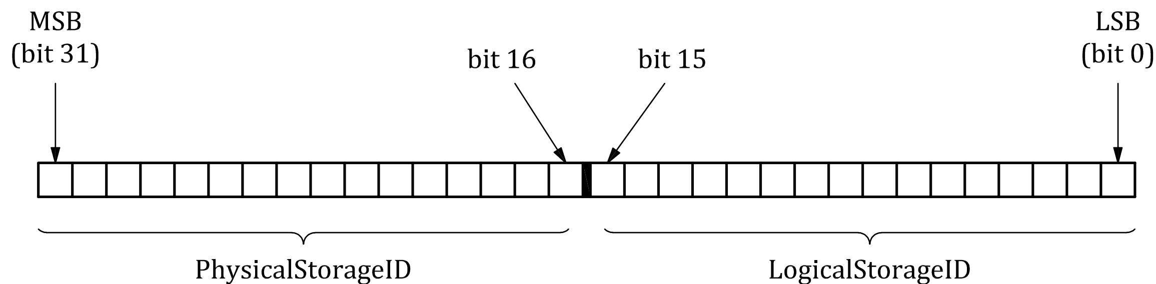

Figure 4 — StorageID Layout

A physical store may possess zero or more logical stores. The StorageIDs for all logical stores within a physical store shall have the same PhysicalStorageID (i.e. the upper 16 bits shall be identical). PhysicalStorageIDs are unique at all times for any particular device. In cases where there is more than one logical store within a physical store, all LogicalStorageIDs (i.e. the lower 16 bits) within that physical store shall be unique. LogicalStorageIDs need only be unique within a physical store, and not globally across all stores. If a physical store does not contain any logical stores, the LogicalStorageID should be set to 0x0000, and it shall be assumed that the store does not contain any data, nor can it receive any data. Neither PhysicalStorageIDs nor LogicalStorageIDs need be consecutive, nor are they interpreted in any other way than as unique identifiers, in the same manner as handles.

StorageIDs should remain unique and persistent over a session. StorageIDs of 0x00000000 and 0xFFFFFFFF are not used to refer to real stores, but have context-specific meanings, such as “all-stores,” or “the default store.” These context-specific extended meanings are indicated in the sections where they are appropriate.

8.2 Data object referencing

8.2.1 Referencing via ObjectHandles

8.2.1.1 Interoperability

In order to ensure interoperability, images and data files shall be referenced using a 4-byte unsigned integer (heretofore referred to as an ObjectHandle). For information on this datatype, see 5.3.3.

8.2.1.2 ObjectHandle assignment

All conforming DSPDs shall allow data object referencing via the following behaviour.

a) ObjectHandles shall be globally unique across all physical and logical storage areas on the device.

b) The values 0x00000000 and 0xFFFFFFFF shall be considered undefined or have context-specific meanings, and shall not be assigned to any valid object.

c) ObjectHandles shall be persistent for each data object present on the device for at least the particular session within which they are exposed, unless the object that the handle refers to becomes inaccessible (e.g. the data object is deleted or the store in which it resides is removed, in which case the device would be required to have issued the appropriate event indicating this change).

d) A particular ObjectHandle is only meaningful when used in the context of the device that assigned that handle. Moving the data object to another device does not imply that it will have the same handle on the new device that is possessed on the originating device, nor the originally assigned handle if the object is placed back in the originating device.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

e) For each new data object that is added to the device while a session is open, the device shall:

   1) immediately assign a unique handle to the newly acquired data object which does not conflict with any other currently assigned handles;

   2) broadcast an appropriate ObjectAdded event to all open sessions in accordance with Clause 12; the initiator is responsible for re-obtaining the StorageInfo data set, if necessary, in order to get an updated version of the FreeSpaceInBytes or FreeSpaceInImages field.

f) If a data object is removed from the device during a session, the device shall broadcast an appropriate ObjectRemoved event to all open sessions in accordance with Clause 12. If the data object is deleted as a result of an operation, the session within which the operation was issued should not receive the event. The initiator is responsible for re-obtaining the StorageInfo data set, if necessary, in order to get an updated version of the FreeSpaceInBytes or FreeSpaceInImages field.

8.2.2 AccessCapability

All stores have an AccessCapability field in their StorageInfo data set which indicates whether they are read-only or not. If a store is read-only, objects cannot be sent to it, regardless of the individual protection status of individual data objects, as described in each object's ObjectInfo data set. Optionally, a device may still allow objects to be deleted from the store even if the store is read-only with regard to new objects being placed there. For a description of permitted values, see 5.5.4. For a description of individual data object protection, refer to the description of the ObjectInfo data set in 5.5.3.

8.3 Receiver object placement

The object receiver shall possess the ability to determine where to place an incoming object that it has requested. In a GetObject scenario, the object receiver is the initiator. In a SendObject scenario, the object receiver is the responder, and therefore the responder shall possess the ability to determine where to put an object that it is receiving if the destination is unspecified by the sender. The destination location chosen by the receiver may or may not be a function of the information that describes the object in its ObjectInfo data set, such as ObjectFormat, size parameters, etc. In a SendObject scenario, the receiver informs the sender of the actual location in which it will be placed upon operation completion using the response parameters from the SendObjectInfo operation. This is to allow for cases where the sender may have to deal with different kinds of receivers, some more capable than others. This allows receiver intervention from a GUI or receiver-side script to allow all incoming objects to be handled in some prescribed or functionally dynamic way, to queue images for printing by placing them in a spool folder, etc. The sender may attempt to request that the object be stored in a specific place on the receiver using the parameters in the SendObjectInfo operation, but the receiver may or may not allow this behaviour. A particular receiver might not provide a structured filesystem, and may store all images in a "flat" area of memory. This type of device would either ignore all association objects or store the associations using some technique other than a filenaming/foldering convention, such as a database.

In SendObject scenarios, a sending device may attempt one of two techniques for specifying object receiver location.

a) The initiator does not specify receiver location for the object being sent. The responder chooses the location, and informs the initiator where the object will be placed in the response to the SendObjectInfo operation.

b) The initiator attempts to specify where the object should be placed on the responder by using the operation parameters of SendObjectInfo. If the responder cannot comply with placing the object there, the SendObjectInfo operation should fail with one of the following responses:

   1) the appropriate access response such as Invalid_StorageID, Invalid_ParentObject, Store_Full, Access_Denied, etc.; in these cases, the initiator may wish to try the SendObjectInfo operation again with an alternate destination;

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

2) Specification_of_Destination_Unsupported; this is used to indicate that the responder does not support initiator-specified destination locations and the initiator should not attempt to specify the destination again.

9 Communication protocol

9.1 Device roles

Rather than assume a host-to-device connection or a peer-to-peer connection, this International Standard refers to only two devices, an initiator and a responder. The initiator is the device that opens the session and initiates operation requests over a suitable transport according to that transport's implementation specification. A responder is a device that responds to operation requests by sending requested data, responses and events. Devices may have the capability of being only an initiator, only a responder or both. A personal computer that detects and configures a USB camera is likely to only be capable of being an initiator, while a USB-only camera may only be capable of being a responder. An IR camera that opens an IrDA connection with a printer and pushes images to it might only be capable of being an initiator, while that corresponding printer might only be capable of being a responder. A digital camera that can send and receive images to and from other digital cameras must be capable of both roles. By assuming the initiator role, a device is assuming added responsibility for device enumeration, transport aggregation (if the device supports multiple PTP conforming transports), controlling the flow of the conversation, and negotiating the transport-specific attributes of the session, all in transport-specific ways. Initiators typically will have some form of graphical user interface that allows thumbnails to be displayed and selected by a user. Devices that are only responders typically will not have such an interface.

9.2 Sessions

9.2.1 Session usage

Sessions are defined as logical connections between two devices, over which ObjectHandles and StorageIDs are persistent.

Sessions are required when requesting handles from a device, as sessions define the minimum persistence period for handle assignments. Sessions do not need to be opened to obtain device capabilities via the GetDeviceInfo operation, but do need to be opened to transfer image and data objects, as well as their descriptors, such as the StorageInfo and ObjectInfo data sets. Any such data sets communicated during a session are considered valid for the duration of that session unless an explicit known event occurs indicating otherwise. All ObjectHandles and data sets other than the DeviceInfo data set shall be considered invalid outside of the session in which they were provided.

A session is considered open as soon as the OpenSession operation sent by the initiator completes with a valid response from the responder. A session is closed when the CloseSession operation request is sent or the transport closes the communications channel, whichever occurs first.

9.2.2 SessionID

Each session shall have a SessionID that consists of one device-unique 32-bit unsigned integer (UINT32). SessionIDs are assigned by the initiator as a parameter to the OpenSession operation and must be non-zero.

9.3 Transactions

9.3.1 Transaction usage

All transactions are considered atomic invocations whose origins can be traced to a single operation request being issued by an initiator to a responder. All transactions are considered synchronous and blocking within a session unless otherwise indicated. Devices that support multiple sessions must be able to keep each session asynchronous and opaque to each other. Asynchronous types of transaction, such as InitiateCapture, are handled by making the operation initiation synchronous. The response to these

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

types of operation indicates only the success or failure of the operation initiation, and asynchronous events are used to handle the communication of new objects becoming available on the device at a later point in time. If an operation request is received which cannot be accommodated due to another previously invoked asynchronous operation still being executed, the device shall indicate that it is busy, using a Device_Busy response, and the initiator is required to re-issue the operation at a later time.

Transactions in this protocol consist of three phases. Depending on the operation, the data phase may not be present. If the data phase is present, data may either be sent from the initiator to the responder (L → R), or from the responder to the initiator (R → L), but never in both directions for the same operation. The operation and response phases are always present. Only one transaction at a time can take place within a session. A transaction may be cancelled by an event. Figure 5 illustrates the sequence of a transaction, with time increasing from left to right:

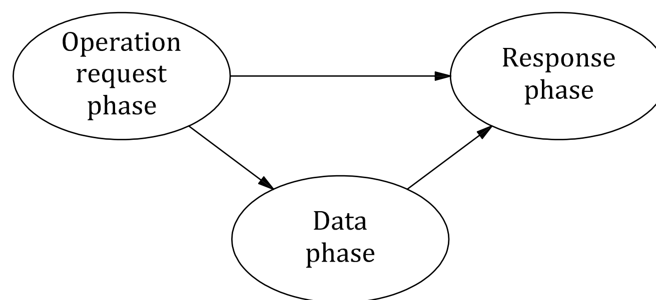

Figure 5 — Transaction sequence

9.3.2 TransactionID

Each transaction within a session shall have a unique transaction identifier called TransactionID that is a session-unique 32-bit unsigned integer (UINT32). TransactionIDs are continuous sequences in numerical order starting from 0x00000001. The TransactionID used for the OpenSession operation shall be 0x00000000. The first operation issued by an initiator after an OpenSession operation shall possess a TransactionID of 0x00000001, the second operation shall possess a TransactionID of 0x00000002, etc. The TransactionID of 0xFFFFFFFF shall not be considered valid, and is reserved for context-specific meanings. The presence of TransactionID allows asynchronous events to refer to specific, previously initiated operations. If this field reaches its maximum value (0xFFFFFFFE), the device should “roll over” to 0x00000001. TransactionIDs allow events to refer to particular operation requests, allow correspondence between data objects and their describing data sets, and aid in debugging.

9.3.3 Operation request phase

The operation request phase consists of the transport-specific transmission of a 30-byte operation data set from the initiator to the responder. Table 20 describes the fields that must be accounted for by the responder in order for the OperationRequest phase to be considered complete.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table 20 — OperationRequest data set

Field	Size (bytes)	Data type
OperationCode	2	UINT16
SessionID	4	UINT32
TransactionID	4	UINT32
Parameter1	4	Any
Parameter2	4	Any
Parameter3	4	Any
Parameter4	4	Any
Parameter5	4	Any

OperationCode: the code indicating which operation is being initiated. For a list of these codes and their usages, see 10.4.

SessionID: the identifier for the session within which this operation is being initiated. This value is assigned by the initiator using the OpenSession operation. This field should be set to 0x00000000 for operations that do not occur within a session, and for the OpenSession OperationRequest data set. See 9.2.2 for a description of SessionIDs.

TransactionID: the identifier of this particular transaction. This value shall be unique within a particular session, and shall increment by one for each subsequent transaction. See 9.3.2 for a description of transaction identifiers. This field should be set to 0x00000000 for the OpenSession operation.

Parameter *n***:** This field holds the operation-specific *n*th parameter. Operations may have at most five parameters. The interpretation of any parameter is dependent upon the OperationCode. Any unused parameter fields should be set to 0x00000000. If a parameter holds a value that is less than 32 bits, the lowest significant bits shall be used to store the value, with the most significant bits being set to zero.

9.3.4 Data phase

The data phase is an optional phase that is used to transmit data that are larger than what can fit into the operation or response data sets. Typically, this is either a data object such as an image, an ancillary data file, or other data sets that are defined in this International Standard or are vendor extensions. All data that are not composed of small, fixed-length types shall be sent via a data phase. For any particular operation, during the data phase, data may flow from the responder to the initiator, from the initiator to the responder or not at all. However, no operation shall send data in both directions during the data phase.

The format for data transferred during a data phase is context-specific by operation and ObjectFormat. If a particular transport wishes to wrap the data with some sort of header, or allow for division into multiple packets with wrappers for re-assembly and possibly event insertion, it is the responsibility of the transport implementation to specify how the data should be handled, wrapped and re-assembled. If the data phase is being used to transport a data object, it shall be considered opaque to the transport. If a data set is being transmitted, it would likely be sent in some sort of transport-specific structure(s) or re-assembled from various transport-specific descriptors.

9.3.5 Response phase

The response phase consists of the transport-specific transmission of a 30-byte response data set from the responder to the initiator. Table 21 lists the fields that are required to be accounted for as part of the response phase.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table 21 — Response data set

Field	Size (bytes)	Format
ResponseCode	2	UINT16
SessionID	4	UINT32
TransactionID	4	UINT32
Parameter1	4	Special
Parameter2	4	Special
Parameter3	4	Special
Parameter4	4	Special
Parameter5	4	Special

ResponseCode: indicates the interpretation of the response as defined in the ResponseCode section in 11.2.

SessionID: the identifier for the session within which this operation is being responded to. This value is assigned by the initiator using the OpenSession operation and should be copied from the OperationRequest data set that is received by the responder prior to responding.

TransactionID: the identifier of the particular transaction. This field should be copied from the OperationRequest data set that is received by the responder prior to responding.

Parameter *n*: This field holds the operation-specific *n*th response parameter. Response data sets may have at most five parameters. The interpretation of any parameter is dependent upon the OperationCode for which the response has been generated, and secondarily may be a function of the particular ResponseCode itself. Any unused parameter fields should be set to 0x00000000. If a parameter holds a value that is less than 32 bits, the lowest significant bits shall be used to store the value, with the most significant bits being set to zero.

9.4 Operation flow

The exact order of operations, responses and events varies depending on the scenario. The initiator determines the flow of operations, while the responder issues responses to the operations and may also issue events. Push scenarios consist of an initiator sending one or more objects to the responder. In Pull scenarios, the initiator retrieves objects from the responder. Pull scenarios usually first involve retrieving thumbnails so that a user can choose which images to retrieve. Ancillary data can also be transferred through either of these modes.

An initiator should always be prepared to receive an event asynchronously relative to response or data phases, as event reception is not limited to transaction phase boundaries. This is of particular concern during the data phase, which may be lengthy if large image or data objects are being transferred. Similarly, the responder should always be prepared to receive an event in place of an operation request, data or a response for transports that use in-band events. See Clause 12 for a description of events.

9.5 Vendor extensions

9.5.1 Default vendor extensions (native)

Device manufacturers can extend this International Standard through several mechanisms outlined in this specification. The VendorExtensionID and VendorExtensionVersion fields of the DeviceInfo data set can be used to uniquely identify the vendor extensions, e.g. two vendors may happen to use the same vendor extended OperationCode 0x8001 for different uses, but the VendorExtensionID and/or VendorExtensionVersion fields for the two devices will be different, and therefore the intended interpretation and usage can be determined. All vendor extended values shall be qualified with the VendorExtensionID for unique identification, e.g. a device cannot use a vendor extended operation until it checks the VendorExtensionID and VendorExtensionVersion fields sent by the other device in the

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

DeviceInfo data set. VendorExtensionIDs shall be assigned by the competent organization as indicated at http://www.iso.org/iso/catalogue_detail?csnumber=63602. VendorExtensionVersion numbers shall be maintained internally by each vendor, and should be publicly advertised if vendors wish others to be able to access their extensions.

9.5.2 Multiple vendor extensions (mapped) PTP v1.1

PTP version 1.1 provides support for multiple vendor extensions using an optional mapping technique. This is specified via two new operations: GetVendorExtensionMaps and GetVendorDeviceInfo. Devices that provide support for multiple vendor-extension sets shall provide support for both of these operations, as well as the new VendorExtensionMap data set. Devices that support both of these operations will interpret vendor extended codes as the default, or “native” set, as indicated by the VendorExtensionID in the standard DeviceInfo data set, in their original vendor-specific opcode. Other sets are supported by calling opcodes “mapped” to the unused portions of the default set, as described by responses to the new operations.

It is the responsibility of the responder to create a mapping of the datacodes of all the supported vendor sets into one combined set. This mapping will be made available to the initiator upon request through an optional GetExtensionsMap operation. The responder will create the mapping in such a way that the datacodes of the default vendor set maintain their values. As a result, initiator implementations based on PTP version 1.0 will be able to communicate with the responder as if it were also based on PTP version 1.0 and supported just one vendor set (the default one).

For example, suppose the device supports two vendor extensions VA and VB:

— VA defines two operation codes, 0x9001 and 0x9002, and one event code, 0xC001;

— VB defines one operation code, 0x9001, one response code, 0xA001, and one event code, 0xC001.

Then the combined data set may consist of the following:

— three operation codes: 0x9001, 0x9002 (unchanged from VA), 0x9003 (corresponding to the code 0x9001 of VB);

— one response code 0xA001 (from VB);

— two event codes 0xC001 (unchanged from VA) and 0xC002 (corresponding to 0xC001 from VB).

For the initiator that is based on PTP version 1.0, the responder will behave as if it only supports the default VendorExtensionID (VA), which is reported in its standard DeviceInfo data set.

Support for a particular mapped vendor extended operation or construct implies support for any dependent vendor-specific constructs as specified in that vendor extension set. For example, if a vendor-extended operation were used to send or get a vendor-extended data set, support for that operation would imply support for reading, writing, and/or interpreting that vendor-extended data set as appropriate to fulfill the specified behaviour of that operation in the context of its originating vendor extension specification.

The version (as defined by the VendorExtensionVersion field in the standard DeviceInfo data set) in context for all codes mapped to a particular distinct alternate set of vendor extensions is specified in the DeviceInfo data set particular to that VendorExtensionID, as returned by the GetVendorDeviceInfo operation, and is therefore statically applicable to all constructs mapped to that particular VendorExtensionID.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

10 Operations

10.1 Operation overview

This clause describes the various operations that are defined in this International Standard, along with their parameters and intended usages. The operations have been defined in order to ensure the following:

— the basic core functionality is easily and uniformly accessible for all devices;

— advanced functionality is exposed in an optional, common way for more capable devices;

— vendor extensibility is accessible via a well-defined mechanism;

— the transport layer and the controlling device are completely abstracted.

10.2 Operation parameters

Many operations have parameters that must be accounted for by all implementations, even if particular values of particular parameters are not supported. All parameters are 32 bits in length. If a particular parameter for a particular operation needs less than 32 bits, the least significant bits shall be used, with the non-used most significant bits being set to zero.

For some operations, parameters act like device properties, as both can modify the way an operation is carried out. In the case of an inconsistency between operation parameters and device properties, the parameter shall take precedence over any device property, but will not cause the device property or the device behaviour to change for any subsequent similar operation requests that do not specify the parameter similarly.

10.3 OperationCode format

OperationCodes are transferred as part of the OperationRequest data set, as described in 9.3.3. All OperationCodes shall take the form of a 16-bit integer, are referred to using hexadecimal notation, and have bit 12 set to 1, and bits 13 and 14 set to 0. All non-defined ResponseCodes having bit 15 set to zero are reserved for future use. If a proprietary implementation wishes to define a proprietary OperationCode, bit 15 should be set to 1.

10.4 OperationCode summary

Table 22 summarizes the operations defined by this International Standard and their corresponding OperationCodes. See Clause 15 for information on which operations are required and which are optional. For details on extending operations, see 9.5.

10.5 Operation descriptions

10.5.1 GetDeviceInfo

OperationCode: 0x1001

Operation Parameter1: none

Operation Parameter2: none

Operation Parameter3: none

Data: DeviceInfo data set

Data direction: R → L

ResponseCode Options: OK, Parameter_Not_Supported

--`,,````,,```,,,``,````,`,-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: returns information and capabilities about the responder device by returning a DeviceInfo data set. This data set is described in 5.5.2. This operation is the only operation that may be issued inside or outside of a session. When used outside a session, both the SessionID and the TransactionID in the OperationRequest data set shall be set to 0x00000000.

See Table 22.

Table 22 — Operation summary

Operation code	Operation name	PTP version
0x1000	Undefined	1.0+
0x1001	GetDeviceInfo	1.0+
0x1002	OpenSession	1.0+
0x1003	CloseSession	1.0+
0x1004	GetStorageIDs	1.0+
0x1005	GetStorageInfo	1.0+
0x1006	GetNumObjects	1.0+
0x1007	GetObjectHandles	1.0+
0x1008	GetObjectInfo	1.0+
0x1009	GetObject	1.0+
0x100A	GetThumb	1.0+
0x100B	DeleteObject	1.0+
0x100C	SendObjectInfo	1.0+
0x100D	SendObject	1.0+
0x100E	InitiateCapture	1.0+
0x100F	FormatStore	1.0+
0x1010	ResetDevice	1.0+
0x1011	SelfTest	1.0+
0x1012	SetObjectProtection	1.0+
0x1013	PowerDown	1.0+
0x1014	GetDevicePropDesc	1.0+
0x1015	GetDevicePropValue	1.0+
0x1016	SetDevicePropValue	1.0+
0x1017	ResetDevicePropValue	1.0+
0x1018	TerminateOpenCapture	1.0+
0x1019	MoveObject	1.0+
0x101A	CopyObject	1.0+
0x101B	GetPartialObject	1.0+
0x101C	InitiateOpenCapture	1.0+
0x101D	StartEnumHandles	1.1+
0x101E	EnumHandles	1.1+

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table 22 *(continued)*

Operation code	Operation name	PTP version
0x101F	StopEnumHandles	1.1+
0x1020	GetVendorExtensionMaps	1.1+
0x1021	GetVendorDeviceInfo	1.1+
0x1022	GetResizedImageObject	1.1+
0x1023	GetFilesystemManifest	1.1+
0x1024	GetStreamInfo	1.1+
0x1025	GetStream	1.1+
All other codes with MSN of 0001	Reserved	
All codes with MSN of 1001	Vendor extended operation code	

10.5.2 OpenSession

OperationCode: 0x1002

Operation Parameter1: SessionID

Operation Parameter2: none

Operation Parameter3: none

Data: none

Data direction: N/A

ResponseCode Options: OK, Parameter_Not_Supported, Invalid_Parameter, Session_Already_Open, Device_Busy

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: causes device to allocate resources, assigns handles to data objects if necessary, and performs any connection-specific initialization. The SessionID will then be used by all other operations during the session. Unless otherwise specified, an open session is required to invoke an operation. If the first parameter is 0x00000000, the operation should fail with a response of Invalid_Parameter. If a session is already open, and the device does not support multiple sessions, the response Session_Already_Open should be returned, with the SessionID of the already open session as the first response parameter. The response Session_Already_Open should also be used if the device supports multiple sessions, but a session with that ID is already open. If the device supports multiple sessions, and the maximum number of sessions is open, the device should respond with Device_Busy.

The SessionID and TransactionID fields of the operation data set should both be set to 0x00000000 for this operation.

10.5.3 CloseSession

OperationCode: 0x1003

Operation Parameter1: none

Operation Parameter2: none

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Operation Parameter3: none

Data: none

Data direction: N/A

ResponseCode Options: OK, Session_Not_Open, Invalid_TransactionID, Parameter_Not_Supported

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: closes the session. Causes device to perform any session-specific clean-up.

10.5.4 GetStorageIDs

OperationCode: 0x1004

Operation Parameter1: none

Operation Parameter2: none

Operation Parameter3: none

Data: StorageIDArray

Data direction: R → L

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Parameter_Not_Supported

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: returns a list of the currently valid StorageIDs. This array shall contain one StorageID for each valid logical store. One StorageID should also be present for each removable medium that is not inserted, which would contain a non-zero PhysicalStorageID and a LogicalStorageID with the value 0x0000.

10.5.5 GetStorageInfo

OperationCode: 0x1005

Operation Parameter1: StorageID

Operation Parameter2: none

Operation Parameter3: none

Data: StorageInfo

Data direction: R → L

ResponseCode Options: OK, Session_Not_Open, Invalid_TransactionID, Access_Denied, Invalid_StorageID, Store_Not_Available, Parameter_Not_Supported

Response Parameter1: none

Response Parameter2: none

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Response Parameter3: none

Description: returns a StorageInfo data set for the particular storage area indicated in the first parameter. This data set is defined in 5.5.4.

10.5.6 GetNumObjects

OperationCode: 0x1006

Operation Parameter1: StorageID

Operation Parameter2: [ObjectFormatCode]

Operation Parameter3: [ObjectHandle of Association for which number of children is desired]

Data: none

Data direction: N/A

ResponseCode options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Invalid_StorageID, Store_Not_Available, Specification_By_Format_Unsupported, Invalid_Code_Format, Parameter_Not_Supported, Invalid_ParentObject, Invalid_ObjectHandle, Invalid_Parameter

Response Parameter1: NumObjects

Response Parameter2: none

Response Parameter3: none

Description: returns the total number of objects present in the store indicated by the first parameter. If the number of objects aggregated across all stores is desired, a StorageID of 0xFFFFFFFF may be used. If a single store is specified, and the store is unavailable because of media removal, this operation should return Store_Not_Available.

By default, this operation returns the total number of objects, which includes both image and non-image objects of all types.

The second parameter, ObjectFormatCode, is optional, and may not be supported. This parameter is used to identify a particular ObjectFormatCode, so that only objects of the particular type will be counted towards NumObjects. If the number of objects of all formats that are images is desired, the value 0xFFFFFFFF may be used. If this parameter is not used, it shall be set to 0x00000000. If the value is non-zero, and the responder does not support specification by ObjectFormatCode, it should fail the operation by returning a ResponseCode with the value of Specification_By_Format_Unsupported.

The third parameter is optional, and may be used to request only the number of objects that belong directly to a particular association. If the third parameter is a valid ObjectHandle for an association, this operation should only return the number of ObjectHandles that exist for objects that are direct children of the association, and therefore only the number of ObjectHandles which refer to objects that possess an ObjectInfo data set with the ParentObject field set to the value indicated in the third parameter. If the number of only those ObjectHandles corresponding to objects in the “root” of a store is desired, this parameter may be set to 0xFFFFFFFF. If the ObjectHandle referred to is not a valid ObjectHandle, the appropriate response is Invalid_ObjectHandle. If this parameter is specified and is a valid ObjectHandle, but the object referred to is not an association, the response Invalid_ParentObject should be returned. If unused, this operation returns the number of ObjectHandles aggregated across the entire device (modified by the second parameter), and the third parameter should be set to 0x00000000.

10.5.7 GetObjectHandles

OperationCode: 0x1007

Operation Parameter1: StorageID

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Operation Parameter2: [ObjectFormatCode]

Operation Parameter3: [ObjectHandle of association for which a list of children is desired]

Data: ObjectHandleArray

Data Direction: R → L

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Invalid_StorageID, Store_Not_Available, Invalid_ObjectFormatCode, Specification_By_Format_Unsupported, Invalid_Code_Format, Invalid_ObjectHandle, Invalid_Parameter, Parameter_Not_Supported, Invalid_ParentObject, Invalid_ObjectHandle

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: returns an array of ObjectHandles present in the store indicated by the StorageID in the first parameter. If an aggregated list across all stores is desired, this value shall be set to 0xFFFFFFFF. Arrays are described in 5.4.

The second parameter is optional, and may or may not be supported. This parameter allows the initiator to ask for only the handles that represent data objects that possess a format specified by the ObjectFormatCode. If a list of handles that represent only image objects is desired, this second parameter may be set to 0xFFFFFFFF. If it is not used, it shall be set to 0x00000000. If the value is non-zero, and the responder does not support specification by ObjectFormatCode, it should fail the operation by returning a ResponseCode with the value of Specification_By_Format_Unsupported. If a single store is specified, and the store is unavailable because of media removal, this operation should return Store_Not_Available.

The third parameter is optional, and may be used to request only a list of the handles of objects that belong to a particular association. If the third parameter is a valid ObjectHandle for an Association, this operation should return only a list of ObjectHandles of objects that are direct children of the Association, and therefore only ObjectHandles that refer to objects that possess an ObjectInfo data set with the ParentObject field set to the value indicated in the third parameter. If a list of only those ObjectHandles corresponding to objects in the “root” of a store is desired, this parameter may be set to 0xFFFFFFFF. If the ObjectHandle referred to is not a valid ObjectHandle, the appropriate response is Invalid_ObjectHandle. If this parameter is specified and is a valid ObjectHandle, but the object referred to is not an association, the response Invalid_ParentObject should be returned. If the third parameter is unused, this operation returns ObjectHandles aggregated across the entire device (modified by the second parameter), and the third parameter should be set to 0x00000000.

10.5.8 GetObjectInfo

OperationCode: 0x1008

Operation Parameter1: ObjectHandle

Operation Parameter2: none

Operation Parameter3: none

Data: ObjectInfo

Data direction: R → L

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Invalid_ObjectHandle, Store_Not_Available, Parameter_Not_Supported

Response Parameter1: none

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Response Parameter2: none

Response Parameter3: none

Description: returns the ObjectInfo data set, as described in 5.5.3. The primary purpose of this operation is to obtain information about a data object present on the device before deciding whether to retrieve that object or its thumbnail with a succeeding GetThumb or GetObject operation. This information may also be used by the caller to allocate memory before receiving the object. Objects that possess an ObjectFormat of type Association do not require a GetObject operation, as these objects are fully qualified by their ObjectInfo data set.

10.5.9 GetObject

OperationCode: 0x1009

Operation Parameter1: ObjectHandle

Operation Parameter2: none

Operation Parameter3: none

Data: DataObject

Data direction: R → L

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Invalid_ObjectHandle, Invalid_Parameter, Store_Not_Available, Parameter_Not_Supported, Incomplete_Transfer

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: retrieves one object from the device. This operation is used for all types of data objects present on the device, including both images and non-image data objects, and should be preceded (although not necessarily immediately) by a GetObjectInfo operation that uses the same ObjectHandle. This operation is not necessary for objects of type Association, as these objects are fully qualified by their ObjectInfo data set. If the store that contains the object being sent is removed during the object transfer, the Incomplete_Transfer response should be used, along with the Store_Removed event.

10.5.10 GetThumb

OperationCode: 0x100A

Operation Parameter1: ObjectHandle

Operation Parameter2: none

Operation Parameter3: none

Data: ThumbnailObject

Data direction: R → L

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Invalid_ObjectHandle, No_Thumbnail_Present, Invalid_ObjectFormatCode, Store_Not_Available, Parameter_Not_Supported

Response Parameter1: none

Response Parameter2: none

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Response Parameter3: none

Description: retrieves the thumbnail from the device that is associated with the ObjectHandle that is indicated in the first parameter.

10.5.11 DeleteObject

OperationCode: 0x100B

Operation Parameter1: ObjectHandle

Operation Parameter2: [ObjectFormatCode]

Operation Parameter3: none

Data: none

Data direction: N/A

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Invalid_ObjectHandle, Object_WriteProtected, Store_Read_Only, Partial_Deletion, Store_Not_Available, Specification_By_Format_Unsupported, Invalid_Code_Format, Device_Busy, Parameter_Not_Supported

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: deletes the data object specified by the ObjectHandle from the device if it is not protected. If the ObjectHandle passed has the value of 0xFFFFFFFF, then all objects on the device shall be deleted. Any write-protected objects shall not be deleted by this operation. If one object is indicated for deletion and it is write-protected, the response code Object_WriteProtected shall be returned. If all objects are indicated for deletion and a subset of the objects are write-protected, only the objects that are not protected shall be deleted, and the response code of Partial_Deletion shall be returned. If the store is read-only without object deletion, the response Store_Read_Only should be returned. If the store is read-only with object deletion, this operation should succeed unless other factors prevent it from succeeding.

The second parameter is optional, and may not be supported. This parameter may only be used if the first parameter is set to 0xFFFFFFFF. This parameter is used to indicate that objects only of the type specified are to be deleted. If this second parameter is also set to 0xFFFFFFFF, then only objects that are images shall be deleted. If it is not used, it shall be set to 0x00000000. If the value is non-zero, and the responder does not support specification by ObjectFormatCode, it should fail the operation by returning a ResponseCode with the value of Specification_By_Format_Unsupported.

If the ObjectHandle indicated in the first parameter is an association, then all objects that are a part of that association (and all descendants of descendants) shall be deleted as well. If only individual items within an association are to be deleted, then individual DeleteObject operations should be issued on each object or sub-association individually.

10.5.12 SendObjectInfo

OperationCode: 0x100C

Operation Parameter1: [Destination StorageID on responder]

Operation Parameter2: [Parent ObjectHandle on responder where object should be placed]

Operation Parameter3: none

Data: ObjectInfo

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Data direction: L → R

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Access_Denied, Invalid_StorageID, Store_Read_Only, Store_Full, Invalid_ObjectFormatCode, Store_Not_Available, Parameter_Not_Supported, Invalid_ParentObject

Response Parameter1: Responder StorageID in which the object will be stored

Response Parameter2: Responder Parent ObjectHandle in which the object will be stored

Response Parameter3: Responder's reserved ObjectHandle for the incoming object

Description: used as the first operation when the initiator wishes to send an object to the responder. This operation sends an ObjectInfo data set from the initiator to the responder. All the fields in this ObjectInfo data set are from the perspective of the initiator, meaning that the StorageID, for example, would be interpreted as the StorageID of the store in which the object resides on the initiator before being sent to the responder. This operation is sent prior to the SendObject operation, described in 10.5.13, in order to inform the responder about the properties of the object that it intends to send later, and to effectively ask whether the object can be sent to the responder. A response of OK infers that the receiver can accept the object, and serves to inform the sender that it may now issue a SendObject operation for the object.

The first parameter is optionally used to indicate the store on the responder in which the object should be stored. If this parameter is specified, and the responder will not be able to store the object in the indicated store, the operation should fail, and the appropriate response, such as Specification_Of_Destination_Unsupported, Store_Not_Available, Store_Read_Only, or Store_Full, should be used. If this parameter is unused, it should be set to 0x00000000, and the responder shall decide in which store to place the object, be that a responder-determined default location, or the location with the most room (or possibly the only location with enough room).

The second parameter is optionally used to indicate where on the indicated store the object should be placed (i.e. the association/folder that the object should become a child of). If this parameter is used, the first parameter shall also be used. If the receiver is unable to place the object as a child of the indicated second parameter, the operation should fail. If the problem with the attempted specification is the general inability of the receiving device to allow the specification of the destination, the response Specification_of_Destination_Unsupported should be sent. This response infers that the initiator should not try to specify a destination location in future invocations of SendObjectInfo, as all attempts at such specification will fail. If the problem is only with the particular destination specified, the Invalid_ObjectHandle or Invalid_ParentObject response should be used, depending on whether the ObjectHandle did not refer to a valid object, or whether the indicated object is a valid object but is not an association. If the root directory of the indicated store is desired, the second parameter should be set to 0xFFFFFFFF. If this parameter is unused, it should be set to 0x00000000, and the responder shall decide where in the indicated store the object is to be placed. If neither the first nor the second parameter is used, the responder shall decide both in which store to place the object as well as where to place it within that store.

If the responder agrees that the object may be sent, it is required to retain this ObjectInfo data set until the next SendObject or SendObjectInfo operation is performed subsequently within the session. If the SendObjectInfo operation succeeds, and the next occurring SendObject operation does not return a successful response, the SendObjectInfo held by the responder shall be retained in case the initiator wishes to re-attempt the SendObject operation for that previously successful SendObjectInfo operation. If the initiator wishes to resend the ObjectInfo data set before attempting to resend the object it may do so. Successful completion of the SendObjectInfo operation conveys that the responder possesses a copy of the ObjectInfo and that the responder has allocated space for the incoming data object. Any response code other than OK indicates that the responder has not retained the ObjectInfo data set, and that the object should not attempt to be sent.

For a particular session, the receiving device shall only retain one ObjectInfo at a time that is the result of a SendObjectInfo operation in the memory. If another SendObjectInfo operation occurs before a SendObject operation, the new ObjectInfo shall replace the previously held one. If this occurs, any storage space or memory space reserved for the object described in the overwritten ObjectInfo data set should be freed before overwriting and allocation of the resources for the new ObjectInfo data set.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

The first response parameter of this operation should be set to the StorageID in which the responder will store the object if it is sent. The second response parameter of this operation should be set to the Parent ObjectHandle of the association of which the object becomes a child. If the object is stored in the root of the store, this parameter should be set to 0xFFFFFFFF.

If the initiator wishes to retain associations and/or hierarchies on the responder for the objects it is sending, then the objects should be sent top down, starting with the highest level of the hierarchy, proceeding in either a depth-first or breadth-first fashion down the hierarchy tree. The initiator shall use the responder's newly assigned ObjectHandle in the third response parameter for the ParentObject that is returned in the SendObjectInfo response as the second operation parameter for a child's SendObjectInfo operation.

10.5.13 SendObject

OperationCode: 0x100D

Operation Parameter1: none

Operation Parameter2: none

Operation Parameter3: none

Data: DataObject

Data direction: L → R

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Store_Full, Store_Not_Available, No_Valid_ObjectInfo, Device_Busy, Parameter_Not_Supported, Incomplete_Transfer

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: used as the second operation when the initiator wishes to send an object to the responder, following the SendObjectInfo operation described in 10.5.12. This operation sends a data object to the device to be written to the responder's store, according to the information in the ObjectInfo data set as transmitted during the most recent SendObjectInfo operation in the same session, and the information indicated by the responder in the response parameters of the SendObjectInfo.

Upon successful completion of this operation, the responder should discard and/or invalidate the initiator's ObjectInfo that the responder held while waiting for that object. If there is no valid ObjectInfo held by the responder, the response No_Valid_ObjectInfo should be returned. Any response other than OK indicates that the SendObject failed for the reason indicated by the response code. In this case, the unassigned ObjectInfo should be retained by the responder in case the initiator wishes to attempt to resend the object for, at most, the duration of the session. If the destination store is removed during object transmission, the Incomplete_Transfer response should be issued along with the StoreRemoved event.

10.5.14 InitiateCapture

OperationCode: 0x100E

Operation Parameter1: [StorageID]

Operation Parameter2: [ObjectFormatCode]

Operation Parameter3: none

Data: none

Data direction: N/A

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Invalid_StorageID, Store_Full, Invalid_ObjectFormatCode, Invalid_Parameter, Store_Not_Available, Invalid_Code_Format, Device_Busy, Parameter_Not_Supported

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: causes the device to initiate the capture of one or more new data objects according to its current device properties, storing the data in the store indicated by the first parameter. If the StorageID is 0x00000000, the object(s) will be stored in a store that is determined by the capturing device. If the particular store specified is not available, or no store is specified and there are no stores available, this operation should return Store_Not_Available.

The capturing of new data objects is an asynchronous operation. This operation may be used to capture images or any type of data that can be fully captured using a single operation trigger. For these types of capture, the length of the capture and the number of objects to capture is known a priori by the responder, as opposed to being dynamically terminable after capture initiation by the initiator. A separate operation, InitiateOpenCapture, described in 10.5.28, can be used to support dynamically controlled captures that are terminable by the initiator.

If the ObjectFormatCode in the second operation parameter is 0x00000000, the device shall capture an image in the format that is the default for the device. A successful response to an InitiateCapture operation indicates the responder's acceptance of the InitiateCapture operation, and not the completion status of the actual object capture, which is indicated using the CaptureComplete event.

As the capture is executed, one or more new data objects should be created on the device. The number of objects to be captured is not specified as part of the InitiateCapture operation, but is determined by the state of the capturing device, and may optionally be set by the initiator using an appropriate DeviceProperty. As each of the newly captured objects becomes available, the responder is required to send an ObjectAdded event to the initiator, indicating the ObjectHandle that is assigned to each, as described in 12.6.2. This ObjectAdded event shall contain the TransactionID of the InitiateCapture operation with which it is associated. If, at any time, the store becomes full, the device shall invoke a Store_Full event, which shall contain the TransactionID of the InitiateCapture operation that failed to cause a new object to be stored. In the case of multiple objects being captured, each object shall be handled separately, so any object captured before the store becomes full should be retained. When all objects have been captured, the responder shall send a CaptureComplete event to the initiator. If the Store_Full event has been issued, the CaptureComplete event should not be issued. If another capture is occurring when this operation is invoked, the Device_Busy response should be used. See Figures 6 and 7.

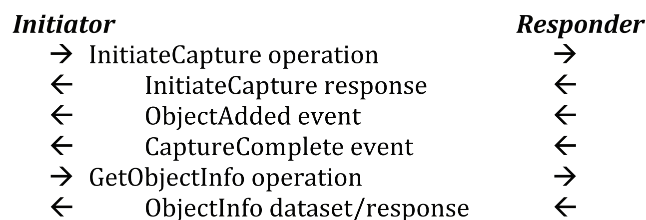

Figure 6 — Single-object InitiateCapture sequence

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

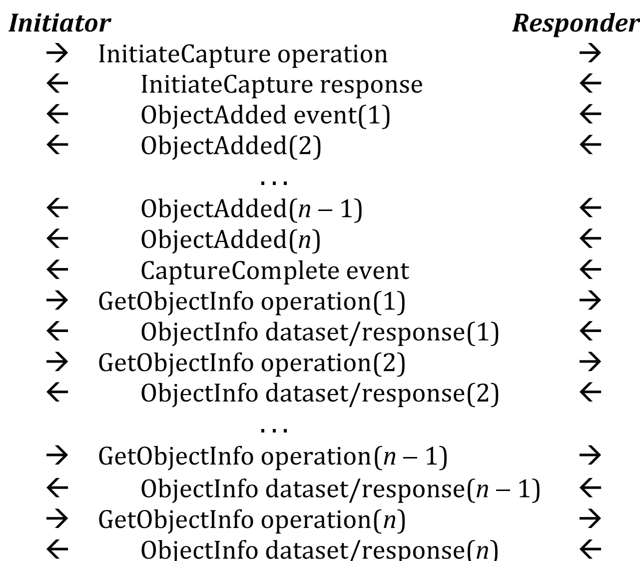

Figure 7 — Multiple-object InitiateCapture sequence

10.5.15 FormatStore

OperationCode: 0x100F

Operation Parameter1: StorageID

Operation Parameter2: [FilesystemType]

Operation Parameter3: none

Data: none

Data direction: N/A

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Invalid_StorageID, Store_Not_Available, Device_Busy, Parameter_Not_Supported, Invalid_Parameter, Store_Read_Only

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: formats the media specified by the StorageID. The second parameter is optional and may be used to indicate the format in which the store should be formatted, according to the FilesystemType codes described in 5.5.4. If a given format is not supported, the response Invalid_Parameter should be returned. If the device is currently capturing objects to the store, or is otherwise unable to format due to concurrent access, the Device_Busy operation should be returned.

10.5.16 ResetDevice

OperationCode: 0x1010

Operation Parameter1: none

Operation Parameter2: none

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Operation Parameter3: none

Data: none

Data direction: N/A

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Device_Busy

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: resets the device to its device-dependent default state. This does not include resetting any device properties, which is performed using ResetDeviceProp. This does include closing the current session, and any other open sessions. If this operation is supported and the device supports multiple concurrent sessions, the device is responsible for supporting the DeviceReset event, which should be sent to all open sessions excluding the one within which the ResetDevice operation was initiated prior to closing the sessions.

10.5.17 SelfTest

OperationCode: 0x1011

Operation Parameter1: [SelfTestType]

Operation Parameter2: none

Operation Parameter3: none

Data: none

Data direction: N/A

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, SelfTest_Failed, Device_Busy, Parameter_Not_Supported

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: causes the device to initiate a device-dependent self-test. The first parameter is used to indicate the type of self-test that should be performed, in accordance with Table 23.

Table 23 — SelfTestType values

Value	Description
0x0000	Default device-specific self-test
All other values with bit 15 set to 0	Reserved
All values with bit 15 set to 1	Vendor-defined

10.5.18 SetObjectProtection

OperationCode: 0x1012

Operation Parameter1: ObjectHandle

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Operation Parameter2: ProtectionStatus

Operation Parameter3: none

Data: none

Data direction: N/A

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Access_Denied, Invalid_ObjectHandle, Invalid_Parameter, Store_Not_Available, Parameter_Not_Supported, Store_Read_Only

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: sets the write-protection status for the data object referred to in the first parameter to the value indicated in the second parameter. For a description of the ProtectionStatus field, refer to the ObjectInfo data set described in 5.5.3. If the ProtectionStatus field does not hold a legal value, the ResponseCode should be Invalid_Parameter.

10.5.19 PowerDown

OperationCode: 0x1013

Operation Parameter1: none

Operation Parameter2: none

Operation Parameter3: none

Data: none

Data direction: N/A

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Device_Busy, Parameter_Not_Supported

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: causes the device to power down. This will cause all currently open sessions to close.

10.5.20 GetDevicePropDesc

OperationCode: 0x1014

Operation Parameter1: DevicePropCode

Operation Parameter2: none

Operation Parameter3: none

Data: DevicePropDesc data set

Data direction: R → L

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Access_Denied, DeviceProp_Not_Supported, Device_Busy, Parameter_Not_Supported

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: returns the appropriate property describing data set as indicated by the first parameter.

10.5.21 GetDevicePropValue

OperationCode: 0x1015

Operation Parameter1: DevicePropCode

Operation Parameter2: none

Operation Parameter3: none

Data: DeviceProperty Value

Data direction: R → L

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, DeviceProp_Not_Supported, Device_Busy, Parameter_Not_Supported

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: returns the current value of a property. The size and format of the data returned from this operation should be determined from the corresponding DevicePropDesc data set returned from the GetDevicePropDesc operation. The current value of a property can also be retrieved directly from the DevicePropDesc, so this operation is not typically required unless a DevicePropChanged event occurs.

10.5.22 SetDevicePropValue

OperationCode: 0x1016

Operation Parameter1: DevicePropCode

Operation Parameter2: none

Operation Parameter3: none

Data: Device Property Value

Data direction: L → R

ResponseCode Options: OK, Session_Not_Open, Invalid_TransactionID, Access_Denied, DeviceProp_Not_Supported, Property_Not_Supported, Invalid_DeviceProp_Format, Invalid_DeviceProp_Value, Device_Busy, Operation_Not_Supported

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: sets the current value of the device property indicated by Parameter1 to the value indicated in the data phase of this operation. The format of the property value object sent in the data phase can be determined from the DatatypeCode field of the property's DevicePropDesc data set. If the property

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

is not settable, the response Access_Denied should be returned. If the value is not allowed by the device, Invalid_DeviceProp_Value should be returned. If the format or size of the property value is incorrect, Invalid_DeviceProp_Format should be returned.

10.5.23 ResetDevicePropValue

OperationCode: 0x1017

Operation Parameter1: DevicePropCode

Operation Parameter2: none

Operation Parameter3: none

Data: none

Data direction: none

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, DeviceProp_Not_Supported, Device_Busy, Parameter_Not_Supported

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: sets the value of the indicated device property to the factory default setting. The first parameter may be set to 0xFFFFFFFF to indicate that all properties should be reset to their factory default settings.

10.5.24 TerminateOpenCapture

OperationCode: 0x1018

Operation Parameter1: TransactionID

Operation Parameter2: none

Operation Parameter3: none

Data: none

Data direction: N/A

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Parameter_Not_Supported, Invalid_Parameter, Capture_Already_Terminated

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: used after an InitiateOpenCapture operation for situations where the capture operation length is open-ended, and determined by the initiator. This operation is not used for trigger captures, which are invoked using a separate operation, InitiateCapture, described in 10.5.14. This operation allows the termination of one capture operation that is being used to capture many objects over a certain period of time, such as a burst, or for long single objects such as manually-controlled image exposures, audio captures, or video clips. The first parameter of this operation indicates the TransactionID of the InitiateOpenCapture operation that is being terminated. If the capture has already terminated for some other reason, this operation should return Capture_Already_Terminated. If the TransactionID parameter

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

does not refer to a transaction that was an InitiateOpenCapture, this operation should return Invalid_TransactionID.

10.5.25 MoveObject

OperationCode: 0x1019

Operation Parameter1: ObjectHandle

Operation Parameter2: StorageID of store to which to move object

Operation Parameter3: ObjectHandle of new ParentObject

Data: none

Data direction: N/A

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Store_Read_Only, Store_Not_Available, Invalid_ObjectHandle, Invalid_ParentObject, Device_Busy, Parameter_Not_Supported, Invalid_StorageID

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: causes the object to be moved from its location within the hierarchy to a new location indicated by the second and third parameters. If the root of the store is desired, the third parameter may be set to 0x00000000. If the third parameter does not refer to a valid object of type Association, the response Invalid_ParentObject should be returned. If a store is read-only (with or without deletion) the response Store_Read_Only should be returned. This operation does not cause the ObjectHandle of the object that is being moved to change.

10.5.26 CopyObject

OperationCode: 0x101A

Operation Parameter1: ObjectHandle

Operation Parameter2: StorageID of the store into which the newly copied object should be placed

Operation Parameter3: ObjectHandle of newly copied object's parent

Data: none

Data direction: N/A

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Store_Read_Only, Invalid_ObjectHandle, Invalid_ParentObject, Device_Busy, Store_Full, Parameter_Not_Supported, Invalid_StorageID

Response Parameter1: ObjectHandle of new copy of object

Response Parameter2: none

Response Parameter3: none

Description: causes the object to be replicated within the responder. The first parameter refers to the ObjectHandle of the object that is to be copied. The second parameter refers to the StorageID into which the newly copied object should be placed. The third parameter refers to the ParentObject of where the newly replicated copy should be placed. If the object is to be copied into the root of the store, this value should be set to 0x00000000.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

10.5.27 GetPartialObject

OperationCode: 0x101B

Operation Parameter1: ObjectHandle

Operation Parameter2: Offset in bytes

Operation Parameter3: Maximum number of bytes to obtain

Data: DataObject

Data direction: R → L

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Invalid_ObjectHandle, Invalid_ObjectFormatCode, Invalid_Parameter, Store_Not_Available, Device_Busy, Parameter_Not_Supported

Response Parameter1: Actual number of bytes sent

Response Parameter2: none

Response Parameter3: none

Description: retrieves a partial object from the device. This operation is optional, and may be used in place of the GetObject operation for devices that support this alternative. If supported, this operation should be generic, and therefore useable with all types of data objects present on the device, including both images and non-image data objects, and should be preceded (although not necessarily immediately) by a GetObjectInfo operation that uses the same ObjectHandle. For this operation, the size fields in the ObjectInfo represent maximum size as opposed to actual size. This operation is not necessary for objects of type Association, as objects of this type are fully qualified by their ObjectInfo data set.

The operation behaves exactly like GetObject, except that the second and third parameters hold, respectively, the offset in bytes and the number of bytes to obtain starting from the offset. If the portion of the object that is desired is from the offset to the end, the third parameter may be set to 0xFFFFFFFF. The first response parameter should contain the actual number of bytes of the object sent, not including any wrappers or overhead structures.

10.5.28 InitiateOpenCapture

OperationCode: 0x101C

Operation Parameter1: [StorageID]

Operation Parameter2: [ObjectFormatCode]

Operation Parameter3: none

Data: none

Data direction: N/A

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Invalid_StorageID, Store_Full, Invalid_ObjectFormatCode, Invalid_Parameter, Store_Not_Available, Invalid_Code_Format, Device_Busy, Parameter_Not_Supported

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Description: causes the device to initiate the capture of one or more new data objects according to its current device properties, storing the data in the store indicated by the StorageID. If the StorageID is 0x00000000, the object(s) will be stored in a store that is determined by the capturing device. If the particular store specified is not available, or no store is specified and there are no stores available, this operation should return Store_Not_Available.

The capturing of new data objects is an asynchronous operation. This operation may be used to implement an Initiate/Terminate mechanism to capture one or more objects over an initiator-controlled time period, such as a single long still exposure, a series of stills, audio capture, etc. Whether the time period controls the time of capture for a single object or the number of fixed-time objects that are captured is determined by the responder, and may be a function of the ObjectFormat as well as any appropriate DeviceProperties.

A separate operation, InitiateCapture, described in 10.5.14, can be used to support captures that do not require the initiator to indicate when the capture should terminate.

If the ObjectFormatCode in the second operation parameter is 0x00000000, the device shall capture an image in the format that is the default for the device. A successful response to an InitiateOpenCapture operation indicates the responder's acceptance of the InitiateOpenCapture operation, and not the completion status of the capture operation.

A successful response to the InitiateOpenCapture operation implies that the responder has started to capture one or more objects. When the initiator wishes to terminate the capture, it is required to send a TerminateOpenCapture operation. The CaptureComplete event is not used for this operation, as the end of the capture period is determined by the initiator. As each of the newly captured objects becomes available, the responder is required to send an ObjectAdded event to the initiator, indicating the ObjectHandle that is assigned to each, as described in 12.6.2. The ObjectAdded event shall contain the TransactionID of the InitiateOpenCapture operation with which it is associated. If, at any time, the store becomes full, the device shall issue a Store_Full event, which shall contain the TransactionID of the InitiateOpenCapture operation that failed to cause a new object to be stored. In the case of multiple objects being captured, each object shall be treated separately, so any object captured before the store becomes full should be retained. Whether or not an object that was partially captured can be retained and used is a function of the device's behaviour and object format; e.g. if the device runs out of room while capturing a video clip, it may be able to save the portion that it had room to store. Any object that is retained in these situations should cause an ObjectAdded event to be issued, while any object that is not retained should cause no event to be issued. A Store_Full event effectively terminates the capture and, in these cases, issuing the TerminateOpenCapture operation is not used. If another object capture is occurring when this operation is invoked, the Device_Busy response should be used.

See Figures 8 and 9.

Initiator		*Responder*
→	InitiateOpenCapture operation	→
←	InitiateOpenCapture response	←
→	TerminateOpenCapture operation	→
←	TerminateOpenCapture response	←
←	ObjectAdded event	←
→	GetObjectInfo operation	→
←	ObjectInfo dataset/response	←

Figure 8 — Single-object InitiateOpenCapture sequence

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

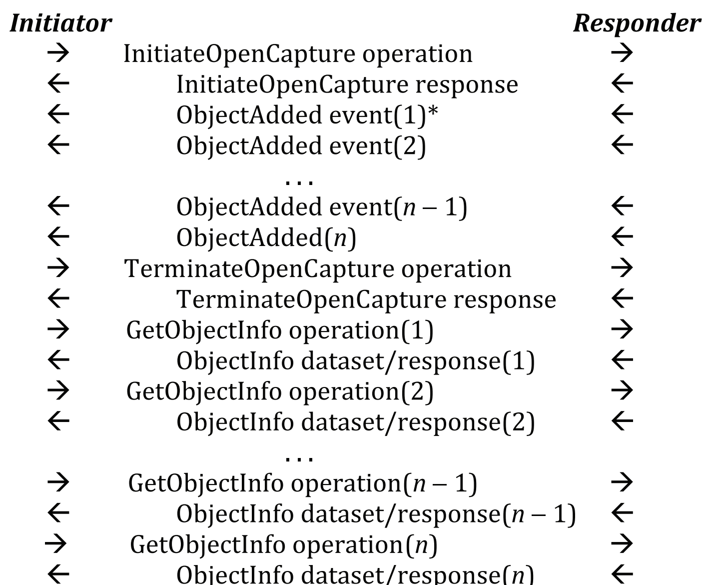

Figure 9 — Multiple-object InitiateOpenCapture sequence

10.5.29 StartEnumHandles

OperationCode: 0x101D

Operation Parameter1: [StorageID]

Operation Parameter2: [ObjectFormatCode]

Operation Parameter3: [ObjectHandle of Association for which a list of children is desired]

Data: none

Data direction: N/A

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Invalid_StorageID, Store_Not_Available, Invalid_ObjectFormatCode, Specification_By_Format_Unsupported, Invalid_Code_Format, Invalid_Parameter, Parameter_Not_Supported, Invalid_ParentObject, Invalid_ObjectHandle, Device_Busy

Response Parameter1: EnumID – UINT32 value

Response Parameter2: none

Response Parameter3: none

Description: Initiates an enumeration process in the scope of the active session. The target storage is indicated by the StorageID. If an aggregated list across all stores is desired, this value shall be set to 0xFFFFFFFF.

The second parameter is optional, and may or may not be supported. This parameter allows the initiator to ask for only the handles that represent data objects that possess a format specified by the ObjectFormatCode. If a list of handles that represent only image objects is desired, this second parameter may be set to 0xFFFFFFFF. If it is not used, it shall be set to 0x00000000. If the value is non-zero, and the responder does not support specification by ObjectFormatCode, it should fail the operation by returning a ResponseCode with the value of Specification_By_Format_Unsupported. If a single store is specified, and the store is unavailable because of media removal, this operation should return Store_Not_Available.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

The third parameter is optional, and may be used to request only a list of the handles of objects that belong to a particular association. If the third parameter is a valid ObjectHandle for an Association, this operation should return only a list of ObjectHandles of objects that are direct children of the Association and therefore only ObjectHandles which refer to objects that possess an ObjectInfo data set with the ParentObject field set to the value indicated in the third parameter. If a list of only those ObjectHandles corresponding to objects in the “root” of a store is desired, this parameter may be set to 0xFFFFFFFF. If the ObjectHandle referred to is not a valid ObjectHandle, the appropriate response is Invalid_ObjectHandle. If this parameter is specified, is a valid ObjectHandle, but the object referred to is not an Association, the response Invalid_ParentObject should be returned.

The first response parameter is used to return an EnumID. This is a unique identifier given to enumerations, so that multiple enumerations can be managed in context. Each specific enumeration, caused by an invocation of StartEnumHandles, has a unique ID that shall not be reused during a session.

10.5.30 EnumHandles

OperationCode: 0x101E

Operation Parameter1: EnumID – UINT32 value

Operation Parameter2: MaxNumberHandles – UINT32 value

Operation Parameter3: None

Data: ObjectHandleArray

Data direction: R → L

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Store_Not_Available, Invalid_Parameter, Invalid_EnumHandle

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: Returns an array of ObjectHandles present in the store indicated at the initiation of the handle enumeration procedure. The first parameter holds the EnumID, which is a unique reference to a set of enumerated ObjectHandles, as returned by the first response parameter of StartEnumHandles described in 10.5.29. The maximum dimension of the array is specified by the second parameter, whose type is UINT32.

Multiple invocations of this operation should never produce repeating values of object handles during the same enumeration procedure.

If there are no more handles, a call to this operation should return an empty ObjectHandleArray. If the number of returned handles is smaller than the one requested (i.e. the enumeration is somewhere at the end), then an ObjectHandleArray should be returned, with a smaller number of elements (the leading UINT32 value of the ObjectHandleArray, indicating the size of the returned array, should contain the number of returned handles).

10.5.31 StopEnumHandles

OperationCode: 0x101F

Operation Parameter1: EnumID

Operation Parameter2: none

Operation Parameter3: none

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Data: none

Data direction: N/A

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Invalid_Parameter, Device_Busy, Invalid_EnumID

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: Closes an active enumeration process (indicated by the EnumID in the first parameter) in the scope of the active session. If no enumeration process is active, return Invalid_EnumID.

10.5.32 GetVendorExtensionMaps

OperationCode: 0x1020

Operation Parameter1: none

Operation Parameter2: none

Operation Parameter3: none

Data: array of VendorExtensionMap data sets

Data direction: N/A

ResponseCode Options: OK, Operation_Not_Supported, Invalid_TransactionID, Device_Busy

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: Used by a PTP v1.1-aware initiator to retrieve the mapping of other vendor specific extensions into the “unused” portion of the current default/native vendor space. In a successful operation, the responder will return an array of VendorExtensionMap data sets. If the responder does not support multiple vendor extensions, it shall return the standard response Operation_Not_Supported for this operation, as well as the related GetVendorDeviceInfo operation.

The format for the array of VendorExtensionMap data sets that is returned as a result of this operation is shown in Table 15. The eight-byte format of each individual VendorExtensionMap data set is specified in Table 24.

NOTE This operation, as well as GetVendorDeviceInfo, can be called outside of a session, in a similar manner to the standard GetDeviceInfo.

Table 24 — Array of VendorExtensionMap data sets (PTP v1.1)

Field	Size (bytes)	Data type
NumberElements	8	UINT64
Element[0]	8	VendorExtensionMap dataset
...	8	VendorExtensionMap dataset
Element[Num – 1]	8	VendorExtensionMap dataset

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

10.5.33 GetVendorDeviceInfo

OperationCode: 0x1021

Operation Parameter1: VendorExtensionID (UINT32)

Operation Parameter2: none

Operation Parameter3: none

Data: DeviceInfo data set for VendorExtensionID specified in Parameter1

Data direction: R → L

ResponseCode Options: OK, Operation_Not_Supported, Invalid_TransactionID, Device_Busy, Unknown_Vendor_Code

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: Used by a PTP v1.1-aware initiator to retrieve the capabilities supported for each vendor. A standard DeviceInfo data set is returned that is particular to the context of the VendorExtensionID specified in the first parameter. If the responder does not support multiple vendor extensions, it shall return the standard response Operation_Not_Supported for this operation, as well as for the related GetVendorExtensionMaps operation. If the VendorExtensionID in the first parameter is not a valid one known to the device, it shall return the standard response Unknown_Vendor_Code. The list of supported VendorExtensionIDs can be built from the information returned by the GetVendorExtensionMaps operation.

NOTE This operation, as well as GetVendorExtensionMaps, can be called outside of a session, in a similar manner to the standard GetDeviceInfo.

10.5.34 GetResizedImageObject

OperationCode: 0x1022

Operation Parameter1: ObjectHandle

Operation Parameter2: image width in pixels

Operation Parameter3: [image height in pixels]

Data: resized ImageObject

Data Direction: R → L

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Invalid_ObjectHandle, Invalid_ObjectFormatCode, Invalid_Parameter, Store_Not_Available, Device_Busy, Parameter_Not_Supported

Response Parameter1: actual number of bytes sent

Response Parameter2: actual width of image sent in pixels

Response Parameter3: actual height of image sent in pixels

Description: An optional operation that retrieves an image object from the device, but typically at a non-standard resolution (i.e. not thumbnail or full resolution). This operation is optional, and may be used in place of the GetObject operation for devices that support this alternative. This operation would be preceded by a GetObjectInfo operation in the same manner as a standard GetObject. The "maximum" image height and width in pixels is stored in the standard ObjectInfo data set.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

As in the standard GetObject operation, the first operation parameter is used to convey the handle of the image object that the operation is being used upon. The second parameter is required, and this is the “new” or “actual” width in pixels of the image that the initiator would like to receive. If the third parameter is unused (i.e. zero), the aspect ratio of the image will not be changed, and the height will be computed by the initiator to be scaled by the same ratio as the image width that was asked for in the second parameter. If the third parameter is specified, the original aspect ratio will be ignored, and the picture will be “forced” to that resolution using a potentially asymmetric “stretch” (no padding/fill or cropping will occur). Optionally, a device may wish to reply with Parameter_Not_Supported if it wishes to refuse asymmetric requests.

10.5.35 GetFilesystemManifest

OperationCode: 0x1023

Operation Parameter1: StorageID

Operation Parameter2: [ObjectFormatCode]

Operation Parameter3: [ObjectHandle of Association of filesystem branch to be characterized]

Data: array of ObjectFilesystemInfo data sets

Data direction: R → L

ResponseCode Options: OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Invalid_ObjectHandle, Invalid_ObjectFormatCode, Invalid_Parameter, Store_Not_Available, Device_Busy, Parameter_Not_Supported

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: This operation retrieves an array of ObjectFilesystemInfo data sets.

The parameters of this operation work in a manner identical to the parameter for the standard GetObjectHandles in every means, with respect to how the parameters operate, filter, etc. The difference is that instead of returning a one-dimensional list of ObjectHandles, the operation returns a newly defined array of ObjectFilesystemInfo data sets. Therefore, use of this operation makes the need to use the standard GetObjectHandles unnecessary.

As in GetObjectHandles, the first parameter represents the store that should be characterized; use of 0xFFFFFFFF specifies “all stores.” Use of the second optional parameter filters the list by ObjectFormatCode, and thus if this parameter is used, only objects of that type will be included in the returned data set array. The third parameter may be used optionally to constrain the returned table to a particular interesting “folder” of the filesystem, like the standard GetObjectHandles that does not include the contents of any subfolders. Both of these parameters default to zero if unused.

ObjectFilesystemInfo data sets are a subset of the ObjectInfo data set, with the addition of a first field that contains the ObjectHandle of the object that the individual following data set describes. The array returned by this operation is specified in Table 25. The ObjectFilesystemInfo data set is specified in Table 16.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table 25 — Array of ObjectFilesystemInfo data sets (PTP v1.1)

Field	Size (bytes)	Data type
NumberElements	8	UINT64
Element[0]	Variable	ObjectFilesystemInfo dataset
...	Variable	ObjectFilesystemInfo dataset
Element[Num – 1]	Variable	ObjectFilesystemInfo dataset

10.5.36 GetStreamInfo

OperationCode: 0x1024

Operation Parameter1: StreamType

Operation Parameter2: none

Operation Parameter3: none

Data: StreamInfo data set

Data Direction: R → L

ResponseCode Options: OK, Operation_Not_Supported, Parameter_Not_Supported, Invalid_Parameter, Session_Not_Open, Invalid_TransactionID

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: This operation is used to retrieve the information about one of the supported streams. The StreamType represents the "type" of one of the streams implemented by the PTP device, as defined in Table 26.

Table 26 — StreamTypes 32-bit bitmask values (PTP v1.1)

Bit Position	StreamType
0 (LSB)	Video
1	Audio
2 to 15	Reserved
16 to 31	Vendor Extended

10.5.37 GetStream

OperationCode: 0x1025

Operation Parameter1: none

Operation Parameter2: none

Operation Parameter3: none

Data: continuous stream of data packets

Data direction: R → L

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

ResponseCode Options: OK, Invalid_TransactionID, Session_NotOpen, Operation_Not_Supported, No_Stream_Enabled

Response Parameter1: none

Response Parameter2: none

Response Parameter3: none

Description: This operation is used after the initiator has prepared the stream(s) for normal operation. This operation will start a continuous stream of packets from the responder to the initiator, holding the responder in a continuous data phase.

The CancelTransaction event may be used at any time the initiator desires to end the streaming, or optionally to “adjust” the properties of the stream(s) in progress. In the latter case, the initiator would cancel the ongoing streaming operation using CancelTransaction, apply the changes (using the new optional stream-related DeviceProperties) and then restart the streaming operation (by re-invoking the new GetStream operation).

11 Responses

11.1 ResponseCode format

ResponseCodes are part of the response data set described in 9.3.5. All ResponseCodes shall take the form of a 16-bit integer, are referred to using hexadecimal notation, and have bits 12 and 14 set to 0 and bit 13 set to 1. All non-defined ResponseCodes having bit 15 set to zero are reserved for future use. If a proprietary implementation wishes to define a proprietary ResponseCode, bit 15 should be set to 1 as well.

ResponseCodes other than “OK” indicate failure of an operation to complete.

11.2 ResponseCode summary

Table 27 summarizes the ResponseCodes defined by this International Standard.

Table 27 — ResponseCode summary

ResponseCode	Description
0x2000	Undefined
0x2001	OK
0x2002	General error
0x2003	Session not open
0x2004	Invalid TransactionID
0x2005	Operation not supported
0x2006	Parameter not supported
0x2007	Incomplete transfer
0x2008	Invalid StorageID
0x2009	Invalid ObjectHandle
0x200A	DeviceProp not supported
0x200B	Invalid ObjectFormatCode
0x200C	Store full
0x200D	Object WriteProtected
0x200E	Store read-only

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table 27 *(continued)*

ResponseCode	Description
0x200F	Access denied
0x2010	No thumbnail present
0x2011	SelfTest failed
0x2012	Partial deletion
0x2013	Store not available
0x2014	Specification by format unsupported
0x2015	No valid ObjectInfo
0x2016	Invalid Code Format
0x2017	Unknown vendor code
0x2018	Capture already terminated
0x2019	Device busy
0x201A	Invalid ParentObject
0x201B	Invalid DeviceProp format
0x201C	Invalid DeviceProp value
0x201D	Invalid parameter
0x201E	Session already open
0x201F	Transaction cancelled
0x2020	Specification of destination unsupported
0x2021	Invalid EnumHandle
0x2022	No stream enabled
0x2023	Invalid Dataset
All other codes with MSN of 0010	Reserved
All codes with MSN of 1010	Vendor extended response code

11.3 Response descriptions

11.3.1 OK

ResponseCode: 0x2001

Description: operation completed successfully.

11.3.2 General error

ResponseCode: 0x2002

Description: operation did not complete. This response is used when the cause of the error is unknown or there is no better failure ResponseCode to use.

11.3.3 Session not open

ResponseCode: 0x2003

Description: indicates that the session handle of the operation is not a currently open session.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

11.3.4 Invalid TransactionID

ResponseCode: 0x2004

Description: indicates that the TransactionID is zero or does not refer to a valid transaction.

11.3.5 Operation not supported

ResponseCode: 0x2005

Description: indicates that the indicated OperationCode appears to be a valid code, but the responder does not support the operation. This error should not normally occur, as the initiator should only invoke operations that the responder indicated were supported in its DeviceInfo data set.

11.3.6 Parameter not supported

ResponseCode: 0x2006

Description: indicates that a non-zero parameter was specified in conjunction with the operation, and that that parameter is not used for that operation. This response is distinctly different from Invalid_Parameter described in 11.3.29.

11.3.7 Incomplete transfer

ResponseCode: 0x2007

Description: indicates that the transfer did not complete. Any data transferred should be discarded. This response should not be used if the transaction was manually cancelled. See the response Transaction_Cancelled, described in 11.3.31.

11.3.8 Invalid StorageID

ResponseCode: 0x2008

Description: indicates that a StorageID sent with an operation does not refer to an actual valid store that is present on the device. The list of valid StorageIDs should be re-requested, along with any appropriate StorageInfo data sets.

11.3.9 Invalid ObjectHandle

ResponseCode: 0x2009

Description: indicates that an ObjectHandle does not refer to an actual object that is present on the device. The list of valid ObjectHandles should be re-requested, along with any appropriate ObjectInfo data sets.

11.3.10 DeviceProp not supported

ResponseCode: 0x200A

Description: the indicated DevicePropCode appears to be a valid code, but that property is not supported by the device. This response should not normally occur, as the initiator should only attempt to manipulate properties that the responder indicated were supported in the DevicePropertiesSupported array in the DeviceInfo data set.

11.3.11 Invalid ObjectFormatCode

ResponseCode: 0x200B

Description: indicates that the device does not support the particular ObjectFormatCode supplied in the given context.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

11.3.12 Store full

ResponseCode: 0x200C

Description: indicates to the store that the operation referred to is full.

11.3.13 Object WriteProtected

ResponseCode: 0x200D

Description: indicates to the object that the operation referred to is write-protected.

11.3.14 Store read-only

ResponseCode: 0x200E

Description: indicates to the store that the operation referred to is read-only.

11.3.15 Access denied

ResponseCode: 0x200F

Description: indicates that access to the data referred to by the operation has been denied. The intent of this response is not to indicate that the device is busy, but that given that the current state of the device does not change, access will continue to be denied.

11.3.16 No thumbnail present

ResponseCode: 0x2010

Description: indicates that a data object exists with the specified ObjectHandle, but the data object does not contain a producible thumbnail.

11.3.17 Self-test failed

ResponseCode: 0x2011

Description: indicates that the device failed an internal device-specific self-test.

11.3.18 Partial deletion

ResponseCode: 0x2012

Description: indicates that only a subset of the objects indicated for deletion has actually been deleted, due to the fact that some were write-protected, or that some objects were on stores that are read-only.

11.3.19 Store not available

ResponseCode: 0x2013

Description: indicates that the store indicated (or the store that contains the indicated object) is not physically available. This can be caused by media ejection. This response shall not be used to indicate that the store is busy, as described in 11.3.25.

11.3.20 Specification by format unsupported

ResponseCode: 0x2014

Description: indicates that the operation attempted to specify action only on objects of a particular format, and that capability is unsupported. The operation should be re-attempted without specifying

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

by format. Any response of this nature infers that any future attempt to specify by format with the indicated operation will result in the same response.

11.3.21 No valid ObjectInfo

ResponseCode: 0x2015

Description: indicates that the initiator attempted to issue a SendObject operation without having previously sent a corresponding SendObjectInfo successfully. The initiator should successfully complete a SendObjectInfo operation before attempting another SendObject operation.

11.3.22 Invalid code format

ResponseCode: 0x2016

Description: indicates that the indicated data code does not have the correct format, and is therefore invalid. This response is used when the Most Significant Nibble of a datacode does not have the format required for that type of code.

11.3.23 Unknown vendor code

ResponseCode: 0x2017

Description: indicates that the indicated data code has the correct format, but has bit 15 set to 1. Therefore, the code is a vendor-extended code, and this device does not know how to handle the indicated code. This response should typically not occur, as the supported vendor extensions should be identifiable by examination of the VendorExtensionID and VendorExtensionVersion fields in the DeviceInfo data set.

11.3.24 Capture already terminated

ResponseCode: 0x2018

Description: indicates that an operation is attempting to terminate a capture session initiated by a preceding InitiateOpenCapture operation, and that the preceding operation has already terminated. This response is only used for the TerminateOpenCapture operation, which is only used for open-ended captures. Subclause 10.5.24 provides a description of the TerminateOpenCapture operation.

11.3.25 Device busy

ResponseCode: 0x2019

Description: indicates that the device is not currently able to process a request because it, or the specified store, is busy. The intent of this response is to imply that perhaps at a future time, the operation should be re-requested. This response shall not be used to indicate that a store is physically unavailable, as described in 11.3.19.

11.3.26 Invalid ParentObject

ResponseCode: 0x201A

Description: indicates that the indicated object is not of type Association, and therefore is not a valid ParentObject. This response is not intended to be used for specified ObjectHandles that do not refer to valid objects, which are handled instead by the Invalid_ObjectHandle response described in 11.3.9.

11.3.27 Invalid DeviceProp format

ResponseCode: 0x201B

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Description: indicates that an attempt was made to set a DeviceProperty, but the DevicePropDesc data set is not the correct size or format.

11.3.28 Invalid DeviceProp value

ResponseCode: 0x201C

Description: indicates that an attempt was made to set a DeviceProperty to a value that is not permitted by the device.

11.3.29 Invalid parameter

ResponseCode: 0x201D

Description: indicates that a parameter was specified in conjunction with the operation, and that although a parameter was expected, the value of the parameter is not a legal value. This response is distinctly different from Parameter_Not_Supported, as described in 11.3.6.

11.3.30 Session already open

ResponseCode: 0x201E

Description: this response code may be used as the response to an OpenSession operation. For multisession devices/transports, this response indicates that a session with the specified SessionID is already open. For single-session devices/transports, this response indicates that a session is open and must be closed before another session can be opened.

11.3.31 Transaction cancelled

ResponseCode: 0x201F

Description: this response code may be used to indicate that the operation was interrupted due to manual cancellation by the opposing device.

11.3.32 Specification of destination unsupported

ResponseCode: 0x2020

Description: this response code may be used as the response to a SendObjectInfo operation to indicate that the responder does not support specification of the destination by the initiator. This response infers that the initiator should not attempt to specify the object destination in any future SendObjectInfo operations, as they will also fail with the same response.

11.3.33 Invalid EnumID

ResponseCode: 0x2021

Description: this response code may be used as the response to an EnumHandles or StopEnumHandles operation, when the indicated EnumID parameter is invalid, such as when there is no active enumeration process (i.e. no previous successful invocation of StartEnumHandles).

11.3.34 No stream enabled

ResponseCode: 0x2022

Description: this response code may be used to indicate that a streaming operation was started without having any streams enabled. This indicates that the initiator should enable one (or more) stream(s) on the responder prior to calling off GetStream operation.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

11.3.35 Invalid data set

ResponseCode: 0x2023

Description: this response code may be used to indicate that a data set is invalid, due to being malformed, having an incorrect size or an invalid field.

NOTE This ResponseCode was referred to in PTP v1.0 but is defined for the first time in PTP v1.1.

12 Events

12.1 Event usage

Although either the initiator or the responder may send an event, most events are typically sent by the responder. The responder also uses events to communicate a state change (e.g. arrival of a new object or store) or optionally to ask the initiator to start a transaction. CancelTransaction is used to cancel a transaction, and may be sent by either an initiator or a responder.

12.2 Event types

12.2.1 Transports with in-band events

For transports that use in-band events due to the lack of existence of a separate logical connection for events, a method of interleaving events into a data stream that meets the device's responsiveness requirements will need to be implemented in a transport-specific fashion.

12.2.2 Transports with out-of-band events

Transports that use a separate logical connection for interrupts effectively have support for out-of-band events. This support means that such transport implementations will not need to break up long image transfers into smaller data blocks in order to accommodate the potential need to interleave events that may occur during the transfer.

12.3 Event data set

See Table 28.

Events are described using the event data set, which consists of the minimal information that is required for qualified notification. Fully qualified events will need to send only this data set in order to fully describe the event and obtain post-event synchronization. In different transport implementations, the fulfillment of all the fields of this data set may or may not happen automatically, and therefore may require a lightweight event notification mechanism separate from the operation of actually requesting the data set fields. Some events, by the very nature of the information that they convey, will infer the need for the event-receiving device to perform an operation to re-synchronize the inter-device state.

Table 28 — Event data set

Field	Size (bytes)	Format
EventCode	2	UINT16
SessionID	4	UINT32
TransactionID	4	UINT32
Parameter1	4	Any
Parameter2	4	Any
Parameter3	4	Any

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

SessionID: indicates the SessionID of the session for which the event is relevant. If the event is relevant to all open sessions, this field should be set to 0xFFFFFFFF. Refer to 9.2.2 for a description of SessionID.

EventCode: indicates the event as defined in the EventCode section.

TransactionID: if the event corresponds to a previously initiated transaction, this field shall hold the TransactionID of that operation. If the event is not specific to a particular transaction, this field shall be set to 0xFFFFFFFF. Refer to 9.3.2 for a description of TransactionID.

Parameter *n***:** this field holds the event-specific *n*th parameter. Events may have at most three parameters. The interpretation of any parameter is dependent upon the EventCode. Any unused parameter fields should be set to 0x00000000. If a parameter holds a value that is less than 32 bits, the Least Significant Bits shall be used to store the value, with the Most Significant Bits being set to zero.

12.4 EventCode format

EventCodes are part of the event data set described in 12.3. All EventCodes shall take the form of a 16-bit integer, are referred to using hexadecimal notation, have bits 12 and 13 set to zero, and bit 14 set to 1. All non-defined ResponseCodes having bit 15 set to zero are reserved for future use. If a proprietary implementation wishes to define a proprietary EventCode, bit 15 should also be set to 1.

12.5 EventCode summary

The events listed in Table 29 are defined in this International Standard.

Table 29 — EventCode summary

EventCode	Name
0x4000	Undefined
0x4001	CancelTransaction
0x4002	ObjectAdded
0x4003	ObjectRemoved
0x4004	StoreAdded
0x4005	StoreRemoved
0x4006	DevicePropChanged
0x4007	ObjectInfoChanged
0x4008	DeviceInfoChanged
0x4009	RequestObjectTransfer
0x400A	StoreFull
0x400B	DeviceReset
0x400C	StorageInfoChanged
0x400D	CaptureComplete
0x400E	UnreportedStatus
All other codes with MSN of 0100	Reserved
All codes with MSN of 1100	Vendor extended response code

12.6 Event descriptions

12.6.1 CancelTransaction

EventCode: 0x4001

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Parameter1: none

Parameter2: none

Parameter3: none

Description: this formal event is used to cancel a transaction for transports that do not have a specified or standard way of cancelling transactions. The particular method used to cancel transactions may be transport-specific. When an initiator or responder receives a CancelTransaction event, it should abort the transaction referred to by the TransactionID in the event data set. If that transaction is already complete, the event should be ignored. After receiving a CancelTransfer event from the initiator, the responder shall send an IncompleteTransfer response for the operation that was cancelled. Both devices will then be ready for the next transaction.

12.6.2 ObjectAdded

EventCode: 0x4002

Parameter1: ObjectHandle

Parameter2: none

Parameter3: none

Description: a new data object has been added to the device. The new handle assigned by the device to the new object should be passed in the Parameter1 field of the event. If more than one object has been added, each new object should generate a separate ObjectAdded event. The appearance of a new store on the device should not cause the creation of new ObjectAdded events for the new objects present on the new store, but should instead cause the generation of a StoreAdded event, as described in 12.6.4.

12.6.3 ObjectRemoved

EventCode: 0x4003

Parameter1: ObjectHandle

Parameter2: none

Parameter3: none

Description: a data object has been removed from the device unexpectedly due to something external to the current session. The handle of the object that has been removed should be passed in the Parameter1 field of the event. If more than one image has been removed, the separate ObjectRemoved events should be generated for each. If the data object that has been removed was removed because of a previous operation that is a part of this session, no event needs to be sent to the opposing device. The removal of a store on the device should not cause the creation of ObjectRemoved events for the objects present on the removed store, but should instead cause the generation of one StoreRemoved event, with the appropriate PhysicalStorageID, as described in 12.6.5.

12.6.4 StoreAdded

EventCode: 0x4004

Parameter1: StorageID

Parameter2: none

Parameter3: none

Description: a new store has been added to the device. If this is a new physical store that contains only one logical store, then the complete StorageID of the new store should be indicated in the first

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

parameter. If the new store contains more than one logical store, then the first parameter should be set to 0x00000000. This indicates that the list of StorageIDs should be re-obtained using the GetStorageIDs operation and examined appropriately. Any new StorageIDs discovered should result in the appropriate invocations of GetStorageInfo operations, as described in 10.5.5.

12.6.5 StoreRemoved

EventCode: 0x4005

Parameter1: StorageID

Parameter2: none

Parameter3: none

Description: the indicated stores are no longer available. The opposing device may assume that the StorageInfo data sets and ObjectHandles associated with those stores are no longer valid. The first parameter is used to indicate the StorageID of the store that is no longer available. If the store removed is only a single logical store within a physical store, the entire StorageID should be sent, which indicates that any other logical stores on that physical store are still available. If the physical store and all logical stores upon it are removed (e.g. removal of an ejectable media with multiple partitions), the first parameter should contain the PhysicalStorageID in the most significant 16 bits, with the least significant 16 bits set to 0xFFFF.

12.6.6 DevicePropChanged

EventCode: 0x4006

Parameter1: DevicePropCode

Parameter2: none

Parameter3: none

Description: a property has changed on the device due to something external to this session. The appropriate property data set should be requested from the opposing device.

12.6.7 ObjectInfoChanged

EventCode: 0x4007

Parameter1: ObjectHandle

Parameter2: none

Parameter3: none

Description: indicates that the ObjectInfo data set for a particular object has changed, and that it should be re-requested.

12.6.8 DeviceInfoChanged

EventCode: 0x4008

Parameter1: none

Parameter2: none

Parameter3: none

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Description: indicates that the capabilities of the device have changed, and that the DeviceInfo should be re-requested. This may be caused by the device going into or out of a sleep state, or by the device losing or gaining some functionality in some way.

12.6.9 RequestObjectTransfer

EventCode: 0x4009

Parameter1: ObjectHandle

Parameter2: none

Parameter3: none

Description: this event can be used by a responder to ask the initiator to initiate a GetObject operation on the handle specified in the first parameter. This allows for push mode to be enabled on devices/transports that are intrinsically pull mode.

12.6.10 Store full

EventCode: 0x400A

Parameter1: StorageID

Parameter2: none

Parameter3: none

Description: this event shall be sent when a store becomes full. Any multi-object capture that may be occurring should retain the objects that were written to a store before the store became full.

12.6.11 Device reset

EventCode: 0x400B

Parameter1: none

Parameter2: none

Parameter3: none

Description: this event needs only to be supported for devices that support multiple sessions or in the case where the device is capable of resetting itself automatically or manually through user intervention while connected. This event shall be sent to all open sessions other than the session that initiated the operation. This event shall be interpreted as indicating that the sessions are about to be closed.

12.6.12 StorageInfoChanged

EventCode: 0x400C

Parameter1: StorageID

Parameter2: none

Parameter3: none

Description: this event is used when information in the StorageInfo data set for a store changes. This can occur due to device properties changing, such as ImageSize, which can cause changes in fields such as FreeSpaceInImages. This event is typically not needed if the change is caused by an in-session operation that affects whole objects in a deterministic manner. This includes changes in FreeSpaceInImages or FreeSpaceInBytes caused by operations such as InitiateCapture or CopyObject, where the initiator can recognize the changes due to the successful response code of the operation, and/or related required events.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

12.6.13 CaptureComplete

EventCode: 0x400D

Parameter1: TransactionID

Parameter2: none

Parameter3: none

Description: this event is used to indicate that a capture session, previously initiated by the InitiateCapture operation, is complete, and that no more ObjectAdded events will occur as a result of this asynchronous operation. This operation is not used for InitiateOpenCapture operations.

12.6.14 UnreportedStatus

EventCode: 0x400E

Parameter1: none

Parameter2: none

Parameter3: none

Description: this event may be implemented for certain transports where situations can arise where the responder was unable to report events to the initiator regarding changes in its internal status. When an initiator receives this event, it is responsible for doing whatever is necessary to ensure that its knowledge of the responder is up to date. This may include re-obtaining individual data sets, ObjectHandle lists, etc., or may even result in the session being closed and re-opened. This event is typically only needed in situations where the transport used by the device supports a suspend/resume/remote-wake-up feature and the responder has gone into a suspend state and has been unable to report state changes during that time period. This prevents the need for queuing of these unreportable events. The details of the use of this event are transport-specific and should be fully specified in the specific transport implementation specification.

13 Device properties

13.1 Device property usage

A device that is conformant to this International Standard may optionally provide different modes of operation or different attributes that can be modified. Collectively these items comprise the device properties. Properties are attributes of the device, and not any particular data object contained in the device. Each property has an associated DevicePropCode. Attributes of individual data objects may be determined by examining their corresponding ObjectInfo data sets as described in 5.5.3, or are contained inside the data objects themselves in a manner specified by its data format, as described by the ObjectFormat field.

13.2 Values of a device property

This International Standard defines how the device is to report the values that it supports for a given property and a means for controlling the setting of the property value. This International Standard also defines how the physical units and the datatype that are the values of a particular property are cast. The device vendor chooses the set of properties to implement.

This International Standard identifies the methods used to:

— determine the value of the factory default setting of a particular device property,

— determine the value of the current setting of a particular device property,

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

— change the current value of a particular device property,

— describe an enumerated list of the properties that the device supports and

— describe the values supported for a given property for a given device.

In a device conforming to this International Standard, the following applies:

— device properties may be read-only or read-write;

— multisession devices shall have one global set of device properties that apply to all sessions; a change in a property caused by one session shall cause a DevicePropChanged event to be issued to all other sessions for each property that changes as a result of the initial change;

— single standard operations are able to be sent or retrieved only one device property at a time; vendor extended properties may be provided that handle multiple properties for one operation;

— property functionality should be able to take advantage of any event mechanism supported by the transport (e.g. a change in a device property value should be able to transparently signal an event to notify any connected devices of a change in state).

13.3 Device property management requirements

This section describes requirements around the management of properties.

A device that is conformant with this International Standard may be requested to perform the following operations.

— The device may be queried (by an initiating device) to determine the properties that it supports.

— The device may be queried (by an initiating device) to determine the possible values for a particular property that it supports.

— The device may be queried (by an initiating device) to determine the current value of a particular property.

— The device may be requested (by an initiating device) to change a current property value to a new value for properties that are identified as being settable.

Upon completion of an operation that requests a change in a device property, other properties on the device may change due to property interdependency. This International Standard does not attempt to provide a mechanism for a priori discovery of these interdependencies. Instead, each operation is limited to directly changing only single properties at a time. If multiple properties are tightly integrated into one multi-dimensional property (e.g. ImageSize, RGBgain), one string property may be used with multiple fields separated by specified delimiters. Each time a property is changed, the device is responsible for changing any other property, in a device-dependent way, that is affected by the new property setting. This typically includes defaulting to a value that is compatible with the new value of the property for which the change was requested. In these cases, the device is required to notify all open sessions with a DevicePropChanged event for each property that changed as a side effect of the requested change.

13.4 Device property identification

13.4.1 Device property behaviour

This International Standard defines a set of DevicePropCodes that identify common properties recognized by all conforming devices capable of functioning as an initiator. A conforming device shall provide information describing what properties it supports. The conforming device shall also provide information that describes the particular values of a property supported by the device. This information is returned to the initiating device when the responding device is queried for the supported values of a particular property.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

13.4.2 Device property describing requirements

The describing information shall contain the DevicePropCode, the datatype of the property, the values of the property supported by the device, and an identification of which of the supported values is the current value as well as the factory default value.

13.4.3 Device property describing methods

A device's supported values of a particular property may be described using one of two methods. The device may return a list that enumerates all supported values of a particular property. Alternatively, in situations where the set of supported values comprises a linear range, a triad of parameters – minimum, maximum and step size – may be used to describe the range of supported property values.

13.4.4 Device property describing data set

A device conforming to this International Standard shall be capable of returning a device property describing data set (DevicePropDesc) for each property supported. This data set is used to hold the information detailing the datatype, permitted values, whether the property is settable, etc. This data set also holds the current value of the property. This data set is returned as the response to the GetDevicePropDesc operation. The GetDevicePropValue operation may also be used if only the current value of the device property is desired.

The supported values of a particular device property may be described by an enumeration of the values a device supports or by describing the minimum and maximum supported range of values with a corresponding inter-value step size. Two forms of the property describing data set are defined in this International Standard to enable the two describing methods. Tables 30 to 32 list the contents of the device property describing data set.

Table 30 — Device property describing data set (DevicePropDesc)

Field	Field order	Size (bytes)	Description
Device property code	1	2	A specific DevicePropCode.
Datatype	2	2	This field identifies the datatype code of the property, as indicated in 5.2.
GetSet	3	1	This field indicates whether the property is read-only (Get) or read-write (Get/Set): 0x00 Get 0x01 Get/Set
Factory default value	4	DTS	This field identifies the value of the factory default setting for the property.
Current value	5	DTS	This field identifies the current value of the property.
Form flag	6	1	This field indicates the format of the next field: 0x00 None. This is for properties like DateTime. In this case the form field is not present. 0x01 Range-Form 0x02 Enumeration-Form
Form	N/A	<variable>	This dataset is the Enumeration-Form or the Range-Form, or is absent if Form Flag = 0.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table 31 — Property describing data set — Range form

Field	Field order	Size (bytes)	Description
MinimumValue	7	DTS	Minimum value of property supported by the device.
MaximumValue	8	DTS	Maximum value of property supported by the device.
StepSize	9	DTS	A particular vendor's device shall support all values of a property defined by: MinimumValue + *n* × StepSize which is less than or equal to MaximumValue where *n* = 0 to a vendor defined maximum.

Table 32 — Property describing data set — Enumeration form

Field	Field order	Size (bytes)	Description
Number of Values	7	2	This field indicates the number of values of size DTS of the particular property supported by the device.
SupportedValue1	8	DTS	A particular vendor's device shall support this value of the property.
SupportedValue2	9	DTS	A particular vendor's device shall support this value of the property.
SupportedValue3	10	DTS	A particular vendor's device shall support this value of the property.
—	—	—	—
SupportedValueM	Special	DTS	A particular vendor's device shall support this value of the property.

13.4.5 DevicePropCode format

All DevicePropCodes shall take the form of a 16-bit integer, are referred to using hexadecimal notation, have bits 12 and 14 set to 1, and bit 13 set to zero. All non-defined DevicePropCodes having bit 15 set to zero are reserved for future use. If a proprietary implementation wishes to define a proprietary DevicePropCode, bit 15 should also be set to 1.

13.4.6 DevicePropCode summary

The properties listed in Table 33 are defined in this International Standard.

Table 33 — DevicePropCode summary

DevicePropCode	Name	PTP version
0x5000	Undefined	1.0+
0x5001	BatteryLevel	1.0+
0x5002	FunctionalMode	1.0+
0x5003	ImageSize	1.0+
0x5004	CompressionSetting	1.0+
0x5005	WhiteBalance	1.0+
0x5006	RGB gain	1.0+
0x5007	F-Number	1.0+
0x5008	FocalLength	1.0+
0x5009	FocusDistance	1.0+

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table 33 *(continued)*

DevicePropCode	Name	PTP version
0x500A	FocusMode	1.0+
0x500B	ExposureMeteringMode	1.0+
0x500C	FlashMode	1.0+
0x500D	ExposureTime	1.0+
0x500E	ExposureProgramMode	1.0+
0x500F	ExposureIndex	1.0+
0x5010	ExposureBiasCompensation	1.0+
0x5011	DateTime	1.0+
0x5012	CaptureDelay	1.0+
0x5013	StillCaptureMode	1.0+
0x5014	Contrast	1.0+
0x5015	Sharpness	1.0+
0x5016	DigitalZoom	1.0+
0x5017	EffectMode	1.0+
0x5018	BurstNumber	1.0+
0x5019	BurstInterval	1.0+
0x501A	TimelapseNumber	1.0+
0x501B	TimelapseInterval	1.0+
0x501C	FocusMeteringMode	1.0+
0x501D	UploadURL	1.0+
0x501E	Artist	1.0+
0x501F	CopyrightInfo	1.0+
0x5020	SupportedStreams	1.1+
0x5021	EnabledStreams	1.1+
0x5022	VideoFormat	1.1+
0x5023	VideoResolution	1.1+
0x5024	VideoQuality	1.1+
0x5025	VideoFrameRate	1.1+
0x5026	VideoContrast	1.1+
0x5027	VideoBrightness	1.1+
0x5028	AudioFormat	1.1+
0x5029	AudioBitrate	1.1+
0x502A	AudioSamplingRate	1.1+
0x502B	AudioBitPerSample	1.1+
0x502C	AudioVolume	1.1+
All other codes with MSN of 0101	Reserved	
All codes with MSN of 1101	Vendor extended property code	

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

13.5 Device property descriptions

13.5.1 BatteryLevel

DevicePropCode = 0x5001

Data type: UINT8

DescForms: Enum, Range

Get/Set: Get

Description: battery level is a read-only property typically represented by a range of integers. The minimum field should be set to the integer used for no power (for example, 0), and the maximum should be set to the integer used for full power (for example, 100). The step field, or the individual thresholds in an enumerated list, are used to indicate when the device intends to generate a DevicePropChanged event to let the opposing device know a threshold has been reached, and therefore should be conservative (for example, 10).

The value 0 may be realized in situations where the device has alternate power provided by the transport or some other means.

13.5.2 FunctionalMode

DevicePropCode = 0x5002

Data type: UINT16

DescForms: Enum

Get/Set: Get, Get/Set

Description: allows the functional mode of the device to be controlled. All devices are assumed to default to a “standard mode”. Alternate modes are typically used to indicate support for a reduced mode of operation (e.g. sleep state) or an advanced mode or add-on that offers extended capabilities. The definition of non-standard modes is device-dependent. Any change in capability caused by a change in FunctionalMode shall be evident by the DeviceInfoChanged event that is required to be sent by a device if its capabilities can change. This property is described using the Enumeration form of the DevicePropDesc data set. This property is also exposed outside of sessions in the corresponding field in the DeviceInfo data set. The permitted values are given in 5.5.2.

13.5.3 ImageSize

DevicePropCode = 0x5003

Data type: String

DescForms: Enum, Range

Get/Set: Get, Get/Set

Description: this property controls the height and width in pixels of the image to be captured that will be supported by the device. This property takes the form of a Unicode, null-terminated string that is parsed as follows: “$W \times H$” where W represents the width and H represents the height interpreted as unsigned integers. Example: width = 800 pixels, height = 600 pixels, ImageSize string = “800 × 600” with a null-terminator on the end. This property may be expressed as an enumerated list of permitted combinations or, if the individual width and height are linearly settable and orthogonal to each other, they may be expressed as a range; e.g. for a device that could set width from 1 pixel to 640 pixels and height from 1 pixel to 480 pixels, the minimum in the range field would be “1 × 1” (null-terminated), for a one-pixel image, and the maximum would be “640 × 480” (null-terminated), for the largest possible

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

image. In this example, the step would be "1 × 1" (null-terminated), indicating that the width and height are each incrementable to the integer.

Changing this device property often causes fields in StorageInfo data sets to change, such as FreeSpaceInImages. If this occurs, the device is required to issue a StorageInfoChanged event immediately after this property is changed.

13.5.4 Compression setting

DevicePropCode = 0x5004

Data type: UINT8

DescForms: Enum, Range

Get/Set: Get, Get/Set

Description: compression setting is a property intended to be as close as is possible to being linear with respect to perceived image quality over a broad range of scene content, and is represented by either a range or an enumeration of integers. Low integers are used to represent low quality (i.e. maximum compression) while high integers are used to represent high quality (i.e. minimum compression). No attempt is made in this International Standard to assign specific values of this property with any absolute benchmark, so any available settings on a device are relative to that device only and are therefore device-specific.

13.5.5 WhiteBalance

DevicePropCode = 0x5005

Data type: UINT16

DescForms: Enum

Get/Set: Get, Get/Set

Description: this property is used to set how the device weights colour channels. The device enumerates its supported values for this property.

See Table 34.

Table 34 — White balance settings

Value	Setting
0x0000	Undefined
0x0001	Manual
0x0002	Automatic
0x0003	One-push automatic
0x0004	Daylight
0x0005	Fluorescent
0x0006	Tungsten
0x0007	Flash
All other values with bit 15 set to zero	Reserved
All values with bit 15 set to 1	Vendor-defined

Manual: The white balance is set directly using the RGB gain property, described in 13.5.6, and is static until changed.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Automatic: The device attempts to set the white balance using some kind of automatic mechanism.

One-push automatic: The user must press the capture button while pointing the device at a white field, at which time the device determines the white balance setting.

Daylight: The device attempts to set the white balance to a value that is appropriate for use in daylight conditions.

Tungsten: The device attempts to set the white balance to a value that is appropriate for use in conditions with a tungsten light source.

Flash: The device attempts to set the white balance to a value that is appropriate for flash conditions.

13.5.6 RGB gain

DevicePropCode = 0x5006

Data type: String

DescForms: Enum, Range

Get/Set: Get, Get/Set

Description: this property takes the form of a unicode, null-terminated string that is parsed as follows: "R:G:B" where the R represents the red gain, the G represents the green gain, and the B represents the blue gain. For example, for an RGB ratio of (red = 4, green = 2, blue = 3), the RGB string could be "4:2:3" (null-terminated) or "2000:1000:1500" (null-terminated). The string parser for this property value should be able to support up to UINT16 integers for R, G and B. These values are relative to each other, and therefore may take on any integer value. This property may be supported as an enumerated list of settings, or using a range. The minimum value would represent the smallest numerical value (typically "1:1:1" null-terminated). Using values of zero for a particular colour channel would mean that that colour channel would be dropped, so a value of "0:0:0" would result in images with all pixel values being equal to zero. The maximum value would represent the largest value each field may be set to (up to "65535:65535:65535" null-terminated), effectively determining the setting's precision by an order of magnitude per significant digit. The step value is typically "1:1:1". If a particular implementation desires the capability to enforce minimum and/or maximum ratios, the green channel may be forced to a fixed value. An example of this would be a minimum field of "1:1000:1", a maximum field of "20000:1000:20000" and a step field of "1:0:1".

13.5.7 F-Number

DevicePropCode = 0x5007

Data type: UINT16

DescForms: Enum

Get/Set: Get, Get/Set

Description: this property corresponds to the aperture of the lens. The units are equal to the F-number scaled by 100. When the device is in an automatic exposure program mode, the setting of this property via the SetDeviceProp operation may cause other properties such as exposure time and exposure index to change. Like all device properties that cause other device properties to change, the device is required to issue DevicePropChanged events for the other device properties that changed as a side effect of the invoked change. The setting of this property is typically only valid when the device has an ExposureProgramMode setting of manual or aperture priority.

13.5.8 FocalLength

DevicePropCode = 0x5008

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Data type: UINT32

DescForms: Enum, Range

Get/Set: Get, Get/Set

Description: this property represents the 35 mm equivalent focal length. The values of this property correspond to the focal length in millimetres multiplied by 100.

13.5.9 FocusDistance

DevicePropCode = 0x5009

Data type: UINT16

DescForms: Enum, Range

Get/Set: Get, Get/Set

Description: the values of this property are unsigned integers with the values corresponding to millimetres. A value of 0xFFFF corresponds to a setting greater than 655 m.

13.5.10 FocusMode

DevicePropCode = 0x500A

Data type: UINT16

DescForms: Enum

Get/Set: Get, Get/Set

Description: the device enumerates the supported values of this property. The values listed in Table 35 are defined.

Table 35 — FocusMode settings

Value	Description
0x0000	Undefined
0x0001	Manual
0x0002	Automatic
0x0003	Automatic macro (close-up)
All other values with bit 15 set to zero	Reserved
All values with bit 15 set to 1	Vendor-defined

13.5.11 ExposureMeteringMode

DevicePropCode = 0x500B

Data type: UINT16

DescForms: Enum

Get/Set: Get, Get/Set

Description: the device enumerates the supported values of this property. The values listed in Table 36 are defined.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table 36 — ExposureMeteringMode settings

Value	Description
0x0000	Undefined
0x0001	Average
0x0002	Centre-weighted-average
0x0003	Multi-spot
0x0004	Centre-spot
All other values with bit 15 set to zero	Reserved
All values with bit 15 set to 1	Vendor-defined

13.5.12 FlashMode

DevicePropCode = 0x500C

Data type: UINT16

DescForms: Enum

Get/Set: Get, Get/Set

Description: the device enumerates the supported values of this property. The values listed in Table 37 are defined.

Table 37 — FlashMode settings

Value	Description
0x0000	Undefined
0x0001	Auto flash
0x0002	Flash off
0x0003	Fill flash
0x0004	Red eye auto
0x0005	Red eye fill
0x0006	External sync
All other values with bit 15 set to zero	Reserved
All values with bit 15 set to 1	Vendor-defined

13.5.13 ExposureTime

DevicePropCode = 0x500D

Data type: UINT32

DescForms: Enum, Range

Get/Set: Get, Get/Set

Description: this property corresponds to the shutter speed. It has units of seconds scaled by 10 000. When the device is in an automatic exposure program mode, the setting of this property via SetDeviceProp may cause other properties to change. Like all properties that cause other properties to change, the device is required to issue DevicePropChanged events for the other properties that changed as a result of the initial change. This property is typically only used by the device when the ProgramExposureMode is set to manual or shutter priority.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

13.5.14 ExposureProgramMode

DevicePropCode = 0x500E

Data type: UINT16

DescForms: Enum

Get/Set: Get, Get/Set

Description: this property allows the exposure program mode settings of the device, corresponding to the "Exposure Program" tag within an Exif or a TIFF/EP image file, to be constrained by a list of permitted exposure program mode settings supported by the device. The values listed in Table 38 are defined.

Table 38 — ExposureProgramMode settings

Value	Description
0x0000	Undefined
0x0001	Manual
0x0002	Automatic
0x0003	Aperture priority
0x0004	Shutter priority
0x0005	Program creative (greater depth of field)
0x0006	Program action (faster shutter speed)
0x0007	Portrait
All other values with bit 15 set to zero	Reserved
All values with bit 15 set to 1	Vendor-defined

13.5.15 ExposureIndex

DevicePropCode = 0x500F

Data type: UINT16

DescForms: Enum, Range

Get/Set: Get, Get/Set

Description: this property allows the emulation of film speed settings on a digital camera. The settings correspond to the ISO designations (ASA/DIN). Typically, a device supports discrete enumerated values but continuous control over a range is possible. A value of 0xFFFF corresponds to the automatic ISO setting.

13.5.16 ExposureBiasCompensation

DevicePropCode = 0x5010

Data type: INT16

DescForms: Enum, Range

Get/Set: Get, Get/Set

Description: this property allows the adjustment of the set point of the digital camera's auto exposure control; e.g. a setting of 0 will not change the factory-set auto-exposure level. The units are in "stops" scaled by a factor of 1 000, in order to allow for fractional stop values. A setting of 2 000 corresponds to 2 stops more exposure (4 × the energy on the sensor), yielding brighter images. A setting of –1 000

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

corresponds to one stop less exposure (1/2 × the energy on the sensor), yielding darker images. The setting values are in APEX units (Additive system of Photographic EXposure). This property may be expressed as an enumerated list or as a range. This property is typically only used when the device has an ExposureProgramMode of the manual.

13.5.17 DateTime

DevicePropCode = 0x5011

Data type: String

DescForms: none

Get/Set: Get, Get/Set

Description: this property allows the current device date/time to be read and set. Date and time are represented in ISO standard format as described in ISO 8601, from the most significant number to the least significant number. This shall take the form of a Unicode string in the format "YYYYMMDDThhmmss.s" where YYYY is the year, MM is the month 01–12, DD is the day of the month 01–31, T is a constant character, hh is the hours since midnight 00–23, mm is the minutes 00–59 past the hour, and ss.s is the seconds past the minute, with the ".s" being optional tenths of a second past the second. This string can optionally be appended with Z to indicate UTC, or ± hhmm to indicate that the time is relative to a time zone. Appending neither indicates that the time zone is unknown.

This property does not need to use a range or an enumeration, as the possible permitted time values are implicitly specified by the definition of standard time and the format given in this and the ISO 8601 specifications.

13.5.18 CaptureDelay

DevicePropCode = 0x5012

Data type: UINT32

DescForms: Enum, Range

Get/Set: Get, Get/Set

Description: this value describes the amount of time delay that should be inserted between the capture trigger and the actual initiation of the data capture. This value shall be expressed in milliseconds. This property is not intended to be used to describe the time between frames for single-initiation multiple captures such as burst or time-lapse, which have separate interval properties outlined in 13.5.25 and 13.5.27. In those cases it would still serve as an initial delay before the first image in the series was captured, independent of the time between frames. For no pre-capture delay, this property should be set to zero.

13.5.19 StillCaptureMode

DevicePropCode = 0x5013

Data type: UINT16

DescForms: Enum

Get/Set: Get, Get/Set

Description: this property allows the specification of the type of still capture that is performed upon a still capture initiation, in accordance with Table 39.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table 39 — StillCaptureMode settings

Value	Description
0x0000	Undefined
0x0001	Normal
0x0002	Burst
0x0003	Timelapse
All other values with bit 15 set to zero	Reserved
All values with bit 15 set to 1	Vendor-defined

13.5.20 Contrast

DevicePropCode = 0x5014

Data type: UINT8

DescForms: Enum, Range

Get/Set: Get, Get/Set

Description: this property controls the perceived contrast of captured images. This property may use an enumeration or range. The minimum supported value is used to represent the least contrast, while the maximum value represents the most contrast. Typically, a value in the middle of the range would represent normal (default) contrast.

13.5.21 Sharpness

DevicePropCode = 0x5015

Data type: UINT8

DescForms: Enum, Range

Get/Set: Get, Get/Set

Description: this property controls the perceived sharpness of captured images. This property may use an enumeration or range. The minimum value is used to represent the least amount of sharpness, while the maximum value represents maximum sharpness. Typically, a value in the middle of the range would represent normal (default) sharpness.

13.5.22 DigitalZoom

DevicePropCode = 0x5016

Data type: UINT8

DescForms: Enum, Range

Get/Set: Get, Get/Set

Description: this property controls the effective zoom ratio of digital camera's acquired image scaled by a factor of 10. No digital zoom (1 ×) corresponds to a value of 10, which is the standard scene size captured by the camera. A value of 20 corresponds to a 2 × zoom where 1/4 of the standard scene size is captured by the camera. This property may be represented by an enumeration or a range. The minimum value should represent the minimum digital zoom (typically 10), while the maximum value should represent the maximum digital zoom that the device allows.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

13.5.23 EffectMode

DevicePropCode = 0x5017

Data type: UINT16

DescForms: Enum

Get/Set: Get, Get/Set

Description: this property addresses special image acquisition modes of the camera. The values listed in Table 40 are defined.

Table 40 — EffectMode setting

Value	Description
0x0000	Undefined
0x0001	Standard (colour)
0x0002	Black and white
0x0003	Sepia
All other values with bit 15 set to zero	Reserved
All values with bit 15 set to 1	Vendor-defined

13.5.24 BurstNumber

DevicePropCode = 0x5018

Data type: UINT16

DescForms: Enum, Range

Get/Set: Get, Get/Set

Description: this property controls the number of images that the device will attempt to capture upon initiation of a burst operation.

13.5.25 BurstInterval

DevicePropCode = 0x5019

Data type: UINT16

DescForms: Enum, Range

Get/Set: Get, Get/Set

Description: this property controls the time delay between captures upon initiation of a burst operation. This value is expressed in whole milliseconds.

13.5.26 TimelapseNumber

DevicePropCode = 0x501A

Data type: UINT16

DescForms: Enum, Range

Get/Set: Get, Get/Set

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Description: this property controls the number of images that the device will attempt to capture upon initiation of a time-lapse capture.

13.5.27 TimelapseInterval

DevicePropCode = 0x501B

Data type: UINT32

DescForms: Enum, Range

Get/Set: Get, Get/Set

Description: this property controls the time delay between captures upon initiation of a time-lapse capture operation. This value is expressed in milliseconds.

13.5.28 FocusMeteringMode

DevicePropCode = 0x501C

Data type: UINT16

DescForms: Enum

Get/Set: Get, Get/Set

Description: this property controls which automatic focus mechanism is used by the device. The device enumerates the supported values of this property. The values listed in Table 41 are defined.

Table 41 — FocusMeteringMode settings

Value	Description
0x0000	Undefined
0x0001	Centre-spot
0x0002	Multi-spot
All other values with bit 15 set to zero	Reserved
All values with bit 15 set to 1	Vendor-defined

13.5.29 UploadURL

DevicePropCode = 0x501D

Data type: String

DescForms: None

Get/Set: Get, Get/Set

Description: this property is used to describe a standard Internet URL (universal resource locator) to which the receiving device may upload images or objects once they are acquired from the device.

13.5.30 Artist

DevicePropCode = 0x501E

Data type: String

DescForms: none

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Get/Set: Get, Get/Set

Description: this property is used to contain the name of the owner/user of the device. This property is intended for use by the device to populate the artist field in any Exif images that are captured with the device.

13.5.31 Copyright

DevicePropCode = 0x501F

Data type: String

DescForms: none

Get/Set: Get, Get/Set

Description: this property is used to contain the copyright notification. This property is intended for use by the device to populate the copyright field in any Exif images that are captured with the device.

13.5.32 SupportedStreams (PTP v1.1)

DevicePropCode = 0x5020

Data type: UINT32

DescForms: none

Get/Set: Get

Description: The responder may implement an arbitrary number of streams, each having a distinct and unique stream-type allocated from 0 to 31. The first 16 stream-types are reserved, while the latter 16 may be used in any vendor-specific fashions. In this context, all current video streams default to a stream-type of 0 and all current audio streams default to a stream type of 1.

This property contains a 32-bit mask. Each bit represents whether a certain stream is supported (stream number bit set to "1") or not (stream number bit set to "0"). Bits 0 to 15 are reserved for standard streams, where currently bit 0 is for video streams and bit 1 is used for audio streams. Bits 16 to 31 are reserved for vendor streams. For example, 0x00000001 means only the video stream is supported, 0x00000003 means video and audio stream are supported. At least one stream should be supported, so 0x00000000 is not a useful value.

This is a mandatory property for streaming. If this property is not supported then the responder does not support streaming at all. Support for this property implies some streaming support (i.e. support for the GetStreamInfo and GetStream operations).

13.5.33 EnabledStreams (PTP v1.1)

DevicePropCode = 0x5021

Data type: UINT32

DescForms: none

Get/Set: Get/Set

Description: This property is used to enable or disable a particular stream or streams. The property holds a 32-bit mask of the same format as for the SupportedStreams property. If a certain bit is set to "0" then the corresponding stream is disabled and should not appear while transferring the streams; if the bit is set to "1" then the corresponding stream is enabled; if the stream is not supported then the bit value has no meaning, e.g. the enabled set of multiplexed streams of the SupportedStreams and EnabledStreams. At least one supported stream should be enabled for streaming in order to function successfully.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

This is a mandatory property for streaming. If this property is not supported then the responder does not support streaming at all. Support for this property implies some streaming support (i.e. support for the GetStreamInfo and GetStream operations and the related stream DeviceProperties).

13.5.34 VideoFormat (PTP v1.1)

DevicePropCode = 0x5022

Data type: UINT32

DescForms: Enum

Get/Set: Get/Set

Description: This DeviceProperty is used to enumerate the supported formats for video streaming and to select a particular format from the supported set.

This DeviceProperty has an enumeration form. Each enumeration value is a four-byte code known as a FOURCC code. FOURCC stands for "Four Character Code" (i.e. coded by four characters, with the first ASCII character in lower byte and the fourth character in the higher byte). For example, the "MJPG" (Motion JPEG) is represented as a 0x47504A4D UINT32 value. Formal specification of FOURCC may be found at the www.fourcc.org website.

The device should enumerate the FOURCC codes in order, from most recommended to the less recommended. The FactoryDefaultValue of the DevicePropDesc data set should be the same as the first value in the enumeration.

The modification of this property often causes changes to the other video properties such as VideoResolution, VideoFrameRate, VideoQuality. If this occurs, the responder is required to issue a DevicePropChanged event. However, it is recommended to keep other properties unchanged if possible and to modify only VideoFrameRate if necessary. The audio stream quality might be altered as well.

This is a mandatory DeviceProperty if video stream is supported.

13.5.35 VideoResolution (PTP v1.1)

DevicePropCode = 0x5023

Data type: String

DescForms: Enum

Get/Set: Get/Set

Description: This DeviceProperty is used to enumerate the supported resolutions for video stream and to select a particular resolution from the supported set.

This DeviceProperty controls the height and width in pixels of the captured video image that will be supported by the device. This property takes the form of a Unicode, null-terminated string that is parsed as follows: "$W \times H$" where W represents the width and H represents the height interpreted as unsigned integers. For example, if the width = 320 pixels and the height = 240 pixels, the VideoResolution string = "320 × 240" with a null-terminator on the end.

This property is expressed as an enumerated list of allowed combinations. The device should enumerate the resolutions in order, from most recommended to least recommended. The FactoryDefaultValue of the DevicePropDesc data set should be the same as the first value in the enumeration.

Modification of this property often causes changes to the other video properties such as VideoFormat, VideoFrameRate, VideoQuality. If this occurs, the responder is required to issue a DevicePropChanged event. However, it is recommended to keep other properties unchanged if possible and to modify only VideoFrameRate if necessary. The audio stream quality might be altered as well.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

This is a mandatory DeviceProperty if video streaming is supported.

13.5.36 VideoQuality (PTP v1.1)

DevicePropCode = 0x5024

Data type: UINT16

DescForms: Enum, Range

Get/Set: Get, Get/Set

Description: This DeviceProperty is used to retrieve or select the quality of the video image.

VideoQuality is a DeviceProperty intended to be as close as possible to being linear with respect to perceived video image quality over a broad range of scene content, and is represented by either a range or an enumeration of integers. Low integers are used to represent low quality (i.e. maximum compression) while high integers are used to represent high quality (i.e. minimum compression). No attempt is made in this standard to assign specific values of this property with any absolute benchmark, so any available settings on a device are relative to that device only and are therefore device-specific.

Modification of this DeviceProperty often causes changes to the other video DeviceProperties such as VideoFormat, VideoFrameRate, VideoResolution. If this occurs, the responder is required to issue a DevicePropChanged event. However, it is recommended to keep other DeviceProperties unchanged if possible and to modify only VideoFrameRate if necessary. The audio stream quality may also be optionally altered.

If this DeviceProperty is read-only it will serve only as an indication of how changing of other properties affects the quality.

This DeviceProperty is recommended but not required if video stream is supported.

13.5.37 VideoFrameRate (PTP v1.1)

DevicePropCode = 0x5025

Data type: UINT32

DescForms: Enum, Range

Get/Set: Get, Get/Set

Description: This DeviceProperty is used to retrieve or select the rate of the video frames. It is represented as an average interval between two video frames, expressed in microseconds.

Modification of this DeviceProperty often causes changes in other video DeviceProperties such as VideoFormat, VideoQuality, VideoResolution. If this occurs, the responder is required to issue a DevicePropChanged event. However, it is recommended to keep other DeviceProperties unchanged if possible and to modify only VideoQuality if necessary. The audio stream quality may be altered as well. If this DeviceProperty is read-only it will serve only as an indication of how changing of other DeviceProperties affect the frame rate.

This is a mandatory DeviceProperty if video streaming is supported.

13.5.38 VideoContrast (PTP v1.1)

DevicePropCode = 0x5026

Data type: UINT16

DescForms: Enum, Range

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Get/Set: Get, Get/Set

Description: This DeviceProperty controls the perceived contrast of video stream. This DeviceProperty may use an enumeration or range. The minimum supported value is used to represent the least contrast, while the maximum value represents the most contrast. Typically, a value in the middle of the range would represent normal (default) contrast.

This is an optional property if video stream is supported.

13.5.39 VideoBrightness (PTP v1.1)

DevicePropCode = 0x5027

Data type: UINT16

DescForms: Enum, Range

Get/Set: Get, Get/Set

Description: This DeviceProperty controls the perceived brightness of video stream. This property may use an enumeration or range. The minimum supported value is used to represent the least brightness, while the maximum value represents the most brightness. Typically, a value in the middle of the range would represent normal (default) brightness.

This is an optional DeviceProperty if video stream is supported.

13.5.40 AudioFormat (PTP v1.1)

DevicePropCode = 0x5028

Data type: UINT32

DescForms: Enum

Get/Set: Get/Set

Description: This property is used to enumerate the supported formats for audio stream and to select a particular format from the supported set.

This property identifies the audio stream format as defined by the fmt chunk in the RIFF (resource interchange file format) file specification. Table 42 summarizes a few values for this property, expanded to 32 bits. For a complete list of properties, refer to RIFF specifications.

Table 42 — Audio format property (PTP v1.1)

Property Code	Name
0x00000001	PCM
0x00000002	ADPCM
0x00000003	IEEE float
0x00000004	VSELP
0x00000005	IBM CVSD
0x00000006	a-Law
0x00000007	u-Law
0x00000008	DTS
0x00000009	DRM
0x00000010	OKI-ADPCM
0x00000011	IMA-ADPCM

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table 42 *(continued)*

Property Code	Name
0x00000012	Mediaspace ADPCM
0x00000013	Sierra ADPCM
0x00000014	G723 ADPCM
0x00000015	DIGISTD
0x00000016	DIGIFIX
0x00000030	Dolby AC2
0x00000031	GSM610
0x0000003b	Rockwell ADPCM
0x0000003c	Rockwell DIGITALK
0x00000040	G721 ADPCM
0x00000041	G728 CELP
0x00000050	MPEG
0x00000052	RT24
0x00000053	PAC
0x00000055	MP3
0x00000064	G726 ADPCM
0x00000065	G722 ADPCM
0x00000101	BM u-Law
0x00000102	IBM a-Law
0x00000103	IBM ADPCM
0x0000674f	OGG VORBIS 1
0x0000676f	OGG VORBIS 1 PLUS
0x00006750	OGG VORBIS 2
0x00006770	OGG VORBIS 2 PLUS
0x00006751	OGG VORBIS 3
0x00006771	OGG VORBIS 3 PLUS
0x0000FFFF	Development
0x0000FFFF-0XFFFFFFFF	Reserved

The device should enumerate the audio format codes in order, from most recommended to least recommended. The FactoryDefaultValue of the DevicePropDesc data set should be the same as the first value in the enumeration.

Modification of this DeviceProperty often causes changes in other audio-related DeviceProperties such as AudioBitRate, AudioSamplingRate. If this occurs the responder is required to issue a DevicePropChanged event.

This is a mandatory DeviceProperty if audio streaming is supported.

13.5.41 AudioBitrate (PTP v1.1)

DevicePropCode = 0x5029

Data type: UINT32

DescForms: Enum

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Get/Set: Get, Get/Set

Description: Theis DeviceProperty is used to enumerate the supported bit rate values for a selected audio format, in bits per second.

This DeviceProperty controls the quality of used compression for the audio stream supported by the device. This property will be an enumeration.

The modification of this DeviceProperty often causes changes in other audio-related DeviceProperties such as AudioFormat and AudioSampleRate. If this occurs the responder is required to issue a DevicePropChanged event.

This is a mandatory DeviceProperty if audio streaming is supported.

13.5.42 AudioSamplingRate (PTP v1.1)

DevicePropCode = 0x502A

Data type: UINT32

DescForms: Enum

Get/Set: Get, Get/Set

Description: This DeviceProperty is used to enumerate the supported sampling rate values for a selected audio format, in hertz.

This DeviceProperty controls the sampling rate used for the audio stream acquisition, supported by the device. This DeviceProperty will be an enumeration. The modification of this DeviceProperty often causes changes to other audio DeviceProperties such as AudioFormat and AudioBitRate. If this occurs the responder is required to issue a DevicePropChanged event.

This is a mandatory DeviceProperty if audio streaming is supported.

13.5.43 AudioBitPerSample (PTP v1.1)

DevicePropCode = 0x502B

Data type: UINT16

DescForms: Enum

Get/Set: Get, Get/Set

Description: This DeviceProperty is used to enumerate the supported sampling rate values for a selected audio format, in hertz.

This DeviceProperty shows/controls the bit per sample used for the audio stream acquisition, supported by the device. This DeviceProperty will be an enumeration. The modification of this DeviceProperty often causes changes in other audio DeviceProperties such as AudioFormat and AudioBitRate. If this occurs the responder is required to issue a DevicePropChanged event.

This is a mandatory DeviceProperty if audio streaming is supported. Note that some compression schemes cannot define a value for AudioBitPerSample, so this property may be zero

13.5.44 AudioVolume (PTP v1.1)

DevicePropCode = 0x502C

Data type: UINT32

DescForms: Enum/Range

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Get/Set: Get, Get/Set

Description: This DeviceProperty is used to represent the volume level of the device, as audio or music is played back to an end-user. This is typically represented by a range of integers, with no volume (audio off) being zero, and maximum volume being the highest integer. This is a relative volume level. A typical implementation might provide 10 or 20 integers that approximate a logarithmic increase in power as to be perceived as an approximately linear increase in volume at a given range.

14 Streaming (PTP v1.1 only)

14.1 Streaming overview

PTP v1.1 specifies an optional mechanism to support streaming content. This clause is concerned with explaining the concepts supported.

Streaming is one of the new features that are supported by more and more digital still cameras (DSC). Streaming support will enhance the user experience by allowing the DSC to be used in video conferencing, and in live preview for taking images, capturing video and audio and any other related actions.

14.2 Stream transfer

The stream transfer is implemented as a standard PTP v1.0 transaction. The operation request data set consists of the new GetStream operation code. The data-in phase is used to transfer the stream data from the responder to the initiator.

It is advised that the initiators perform the GetStream operation in a loop, with a known, large enough data size (e.g. 0x7FFFFFFF), in order to deal with transports which cannot deal with unknown data size (e.g. USB). For transports that do support unknown data size (e.g. PTP-IP), the GetStream operation should use the transport-defined unknown data size (e.g. for PTP-IP this is 0xFFFFFFFF).

The data phase normally does not end by itself and must be explicitly cancelled by the initiator. Thus, by issuing the GetStream operation, the initiator and responder will enter the streaming mode and the stream data will be transferred over the data-in phase in an infinite fashion. The initiator may need to cancel the transaction in two cases: when it should terminate the streaming or when it wishes to adjust on-fly some stream properties (e.g. brightness, contrast). In the latter case the initiator will re-issue the GetStream operation after changing the properties. The responder may enhance this situation by internally buffering the stream data, in order to avoid stream gaps that might occur while changing the on-fly properties. The responder shall be able to cancel the streaming operation in the same manner as for other standard PTP v1.0 operations (i.e. issuing a CancelTransaction event and/or using a transport-specific mechanism as appropriate, as defined in that transport implementation specification).

14.3 Multiplexing

The multiplexing approach is introduced to allow multiple stream transfer such as video and audio over a single PTP data channel. As shown in Figure 10, the streams generated by the responder are divided into stream packets and transmitted over PTP data-in. The initiator receives the stream packets and re-assembles the original streams. The order and the size of the packets are controlled by the responder in such a manner as to maintain the real-time human perception of video and audio, as appropriate. Each packet is marked with the stream number. Neither the responder nor the initiator is required to support multiple streams; the simplest supported case is when there is one video stream. In the case of a single video stream, all the packets will belong to the same stream. Besides stream packets there are also stream frames. While packets are used as multiplexing units and are no longer useful after the initiator reassembles the streams, the frames are used for further stream processing and their payload is specific to a particular stream format.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

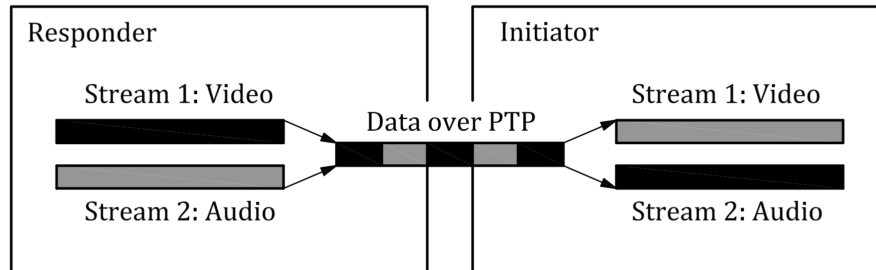

Figure 10 — Using multiplexing for stream transfer

14.4 Discovering and configuring stream capabilities

Before performing the stream acquisition, the initiator should negotiate stream capabilities. This is performed using DeviceProperties and includes the following tasks:

— discovering what kind of streams the responder supports;

— enabling or disabling the particular stream;

— selecting the stream properties such as format, resolution, frame rate, etc.;

— retrieving the information used to handle the stream such as timer resolution, packet alignment, maximal frame size, etc.

A typical initiator will tend to follow the order described above in order to obtain optimal frame size, which may require selecting a particular format, resolution, quality or frame rate. Many of these types of property are interrelated, and may undergo changes as a result of other properties being changed (e.g. increasing the frame rate might cause a certain responder to automatically decrease the resolution to compensate for bandwidth limitations). Such indirect changes should be accomplished via the standard DevicePropChanged event system.

14.5 Data transfer mechanism

This subclause explains the data transfer mechanism that is employed by the streaming support. After all of the initial steps (discover, enable, select properties and determine stream-specific information) have been performed, the stream acquisition process may begin. This is initiated by invoking the new GetStream operation.

Most streaming technologies are based upon frames. The frame, in the context of this specification, is the minimum indivisible data-unit specific to a particular stream-format, and it contains streaming-specific information. However, in order to allow the transfer of multiple streams, the frame must be broken into a number of packets. Before this, the frame has some extra headers appended which may be used by the initiator to reproduce the stream (such as timing sync info, etc...).

A frame (especially the lengthy ones, i.e. video frames) can be divided across multiple packets for delivery, from responder to initiator. The packets offer the basic mechanism for the responder to multiplex multiple streams. The packets will contain the StreamID, so the initiator will be able to reconstruct the original frames for each of the involved streams. This is described in Figure 11.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

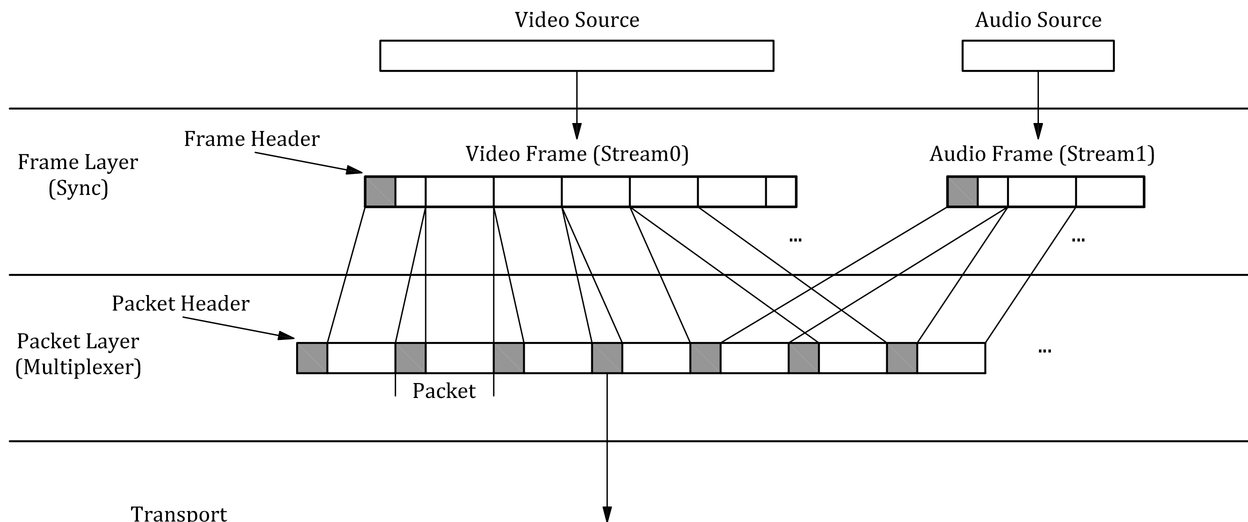

Figure 11 — Streaming data transfer mechanism

14.6 Packet layout

The packet layout is presented in Figure 12.

Flags	Stream type	Sequence No.	Payload Size	Payload
8 bits	8 bits	16 bits	16 bits	variable

Packet Layout

Time Stamp	Frame Size	Payload
32 bits	32 bits	variable

Frame Layout

Figure 12 — Packet and frame layout

The fields have the following signification:

— **Flags** can be represented as the type of the packet. The packet can be first, normal packet or last packet (0x00, 0x01 and 0x02).

— **StreamType** field contains the StreamType that the packet belongs to (i.e. bit zero represents video and bit one reprents audio). StreamTypes are formally defined in Table 26.

— **Sequence No.** field represents the order of the packet in the frame. It is used by the initiator to understand if, for any reason, packets are lost.

— **Payload Size** field contains the size in bytes of the packet payload.

— **Payload** contains a variable number of bytes (indicated by the payload size field) from the upper layer (frame).

No CRC or checksum is required since there is no retransmission mechanism involved. However, the data integrity check may be provided by the specific video or audio formats at their own frame level.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

14.7 Frame layout

The frame layout is presented in Figure 12.

— **Time Stamp** represents the current timer value of the responder. In combination with the timer resolution value (obtained through the stream info. data set), this will give the initiator the timing information, in absolute units, required for time synchronization.

— **Frame Size** represents the total number of bytes that the frame contains.

— **Payload** represents stream-specific data.

14.8 Enumerating supported streams

The new SupportedStreams DeviceProperty is used to obtain the list of streams the responder supports. This DeviceProperty is described in 13.5.32.

14.9 Retrieving stream information

As a next logical step after selecting stream properties the initiator must typically retrieve the information for each stream. This information is used to perform stream acquisition.

The StreamInfo data set is described in Table 17 and is retrieved using the new optional GetStreamInfo operation, described in 10.5.36.

15 Conformance section

To determine if the responder supports push mode, the initiator should look for the SendObject OperationCode in the device's DeviceInfo data set. Presence of the GetObject operation indicates pull mode is supported.

An initiator shall be able to send operations, receive responses and events, and poll for in-band events, if necessary, for that transport. A responder shall be able to respond to all operations that it reports in its DeviceInfo data set. The specific operations a particular device is required to support are listed in Table 43.

Table 43 — Operation implementation conformance

OperationCode	**Operation name**	**PTP v1.0**		**PTP v1.1**	
		Pull	Push	Pull	Push
0x1001	GetDeviceInfo	**M**	**M**	**M**	**M**
0x1002	OpenSession	**M**	**M**	**M**	**M**
0x1003	CloseSession	**M**	**M**	**M**	**M**
0x1004	GetStorageIDs	**M**	**M**	**M**	**M**
0x1005	GetStorageInfo	**M**	**M**	**M**	**M**
0x1006	GetNumObjects	**M**	O	**M**	O
0x1007	GetObjectHandles	**M**	O	**M**	O
0x1008	GetObjectInfo	**M**	O	**M**	O
0x1009	GetObject	**M**	O	**M**	O
0x100A	GetThumb	**M**	O	**M**	O
0x100B	DeleteObject	O	O	O	O
0x100C	SendObjectInfo	O	**M**	O	**M**
0x100D	SendObject	O	**M**	O	**M**
NOTE **M**: mandatory; O: optional; *N/A*: not available.					

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table 43 *(continued)*

OperationCode	Operation name	PTP v1.0		PTP v1.1	
		Pull	Push	Pull	Push
0x100E	InitiateCapture	O	O	O	O
0x100F	FormatStore	O	O	O	O
0x1010	ResetDevice	O	O	O	O
0x1011	SelfTest	O	O	O	O
0x1012	SetObjectProtection	O	O	O	O
0x1013	PowerDown	O	O	O	O
0x1014	GetDevicePropDesc	O	O	O	O
0x1015	GetDevicePropValue	O	O	O	O
0x1016	SetDevicePropValue	O	O	O	O
0x1017	ResetDevicePropValue	O	O	O	O
0x1018	TerminateOpenCapture	O	O	O	O
0x1019	MoveObject	O	O	O	O
0x101A	CopyObject	O	O	O	O
0x101B	GetPartialObject	O	O	O	O
0x101C	InitiateOpenCapture	O	O	O	O
0x101D	StartEnumHandles	*N/A*	*N/A*	O	O
0x101E	EnumHandles	*N/A*	*N/A*	O	O
0x101F	StopEnumHandles	*N/A*	*N/A*	O	O
0x1020	GetVendorExtensionMaps	*N/A*	*N/A*	O	O
0x1021	GetVendorDeviceInfo	*N/A*	*N/A*	O	O
0x1022	GetResizedImageObject	*N/A*	*N/A*	O	O
0x1023	GetFilesystemManifest	*N/A*	*N/A*	O	O
0x1024	GetStreamInfo	*N/A*	*N/A*	O	O
0x1025	GetStream	*N/A*	*N/A*	O	O
NOTE **M**: mandatory; O: optional; *N/A*: not available.					

A device shall be capable of generating and/or responding to all events as indicated in Table 44. A responder may not need to issue many events if it disables manual or external object creation and device property manipulation while a session is open. While a session is open, a responder shall send an event whenever an image, ancillary data or a store is added or removed and when any device property or the DeviceInfo capabilities are modified.

Table 44 — Event implementation conformance

EventCode	Event name	Required?
0x4001	CancelTransaction	Only if transport implementation specification specifies using this formal event for cancelling transactions.
0x4002	ObjectAdded	Only if objects can be added from an external source during a session.
0x4003	ObjectRemoved	Only if objects can be removed by external sources during a session.
0x4004	StoreAdded	Only if device has stores that may become available/unavailable during a session.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table 44 *(continued)*

EventCode	Event name	Required?
0x4005	StoreRemoved	Only if device has stores that may become available/ unavailable during a session.
0x4006	DevicePropChanged	Only if device properties are supported and some properties are inter-dependent or can be changed due to an external or indirect source.
0x4007	ObjectInfoChanged	Only if it is possible for the objects' ObjectInfo to change during a session.
0x4008	DeviceInfoChanged	Only if device supports changes in functionality during an open session, such as sleep modes or functional modes.
0x4009	RequestObjectTransfer	Only if this functionality is desired.
0x400A	StoreFull	Only if objects can be sent to the device or it supports InitiateCapture or InitiateOpenCapture.
0x400B	DeviceReset	Only if it is possible for the device to be reset by something other than the initiator.
0x400C	StorageInfoChanged	Only if it is possible for one of the fields in StorageInfo to change (e.g. FreeSpaceInImages due to an ImageSize DeviceProperty change).
0x400D	CaptureComplete	Only if InitiateOpenCapture is supported.
0x400E	UnreportedStatus	Only if the transport implementation supports suspend without self-resume capability and other events may occur during a suspend period.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Annex A
(informative)

Optional device features

A.1 Scope

This annex describes the features that may be implemented at the vendor's discretion. These features are not required in order that a device conform to this International Standard.

A.2 Open write store

Devices are not required to possess the ability to write images or other data directly to their internal storage areas via an external data transfer. Devices that are read-only may still fully conform to this International Standard. Devices might only be able to record images via a manual press of the shutter button or as a response to an InitiateCapture or InitiateOpenCapture operation.

A.3 Hierarchical storage system

Devices are not required to possess anything other than a flat-filesystem for storing image data, but should be able to take advantage of a hierarchical filesystem if present.

A.4 Multisession capability

Although devices are not required to support more than one simultaneous physical or logical session, they are not explicitly prohibited from doing so by this International Standard. A particular device may effectively deny concurrent service. The current version of this International Standard provides no method for avoiding the pitfalls of data corruption caused by multisession operation.

A.5 Out-of-session image handle persistence

According to 8.2.1.2, devices shall keep assigned image handles persistent within a particular session. However, devices may optionally wish to extend this persistence in the following ways.

a) Powered handle persistence

   Devices that exhibit powered handle persistence retain the same ObjectHandles across all connection sessions while the device is actively powered, only re-enumerating image handles on power-up or when specifically requested to do so.

b) Permanent handle persistence

   Devices that exhibit permanent handle persistence retain the same ObjectHandles permanently, across all connection sessions and power-on sessions, only re-enumerating image handles when specifically requested to do so.

A.6 Multiple image formats

Devices shall support image formats as described in Clause 6. Devices may or may not support multiple image formats, bit depths, heights, widths and aspect ratios.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

A.7 Multiple thumbnail formats

Devices shall support thumbnail formats as described in 6.2. Devices may or may not provide multiple thumbnail formats, bit depths, heights, widths and aspect ratios.

A.8 Write-protection mechanism

Devices may optionally possess a write-protection mechanism for the images that they contain, but they are not required to do so.

A.9 Sub-image transfers

Devices may optionally transfer images subsampled on the fly, to a lower resolution, or may allow specification by subsection. These features are not supported by the standard GetObject operation, but could be enabled by using a vendor-extended operation.

A.10 Non-standard functional modes

Devices may optionally take on non-standard modes of operation such as sleep modes, or advanced functionality modes. A change in mode either requires all sessions to be closed, or requires support for the DeviceInfoChanged event, to indicate the changes in supported capabilities while in the new mode.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Annex B
(normative)

Object referencing and format codes

B.1 Scope

This annex describes the conventions for how ObjectFormats should be implemented by both the initiator (in this case the sender or host computer) and the responder (in this case the receiver or PTP camera).

B.2 Object persistence and referencing

PTP does not assume that a filesystem is internally implemented for either initiators or responders. ObjectHandles were implemented to remove any filesystem dependence. The issues that are being avoided by using this technique are the following:

a) there is no standard filesystem; some filesystems are limited to 8.3 length, ASCII characters, non-UNICODE, etc.; attempting to keep a filename valid between systems is very difficult and insufficient;

b) some devices do not need to store images, and would rather store them transiently in RAM until they are done with them (e.g. printer);

c) some digital cameras may not wish to implement the full burden of a filesystem (e.g. tethered devices);

d) filesystems are generally not as powerful as databases, of which advanced systems may want to take advantage.

When sending objects to another device, specification of the filename field in the ObjectInfo data set is optional.

The filesystem for each device is "hidden" from the opposing device. As a result, the receiving device may not refuse the request based upon any insufficiency in the proposed filename, such as the following:

e) the filename is not specified at all;

f) the filename conflicts with a file already in that location on the receiver, for devices that support specification of destination by the sender;

g) the filename is too long;

h) the filename is not in ASCII.

If the receiving device uses a filesystem for its internal persistent store, it shall create a filename that is suitable for its own local use if any of the above conditions apply. In general, filesystems should be viewed as local to each device. Therefore, if a sender uses a filesystem, and it sends the file to the receiver using a SendObject/Info operation pair, it should query the new object after it is sent, in order to receive the filename the receiver applied to it, if it wishes to view the receiver in a filesystem-like view.

B.3 ObjectFormatCode assignment

PTP uses the ObjectFormatCode field of the ObjectInfo data set in order to convey what type of data are inside an object. This is done in order to remove the filename dependencies upon which some platforms rely. Some other platforms use special areas inside the objects to identify object types. Both receivers and senders should attempt to determine object type according to whatever local mechanism is used to identify these items, knowing that the actual filenames and/or objects themselves cannot always be relied upon.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

B.4 ObjectFormatCode assignment based on typical extensions

Table B.1 shows a mapping from filename extension to ObjectFormatCode, for those systems that need to generate ObjectFormatCodes from filename extensions.

Table B.1 — Summary of ObjectFormatCodes

Information known about object	Format	Object FormatCode
Device Script	Script	0x3002
Device Executable	Executable	0x3003
filename = “Autprint.mrk”	DPOF	0x3006
filename extension = “jpg” or “jpeg”	Exif/JPEG	0x3801
filename extension = “tif” or “tiff”	TIFF	0x380D
filename extension = “gif”	GIF	0x3807
filename extension = “fpx”	FlashPix	0x3803
filename extension = “jp2”	JP2	0x380F
filename extension = “jpx”	JPX	0x3810
filename extension = “bmp”	BMP	0x3804
filename extension = “png”	PNG	0x380B
filename extension = “cif” or “ciff”	CIFF	0x3805
filename extension = “pcd”	PCD	0x3809
filename extension = “pic” or “pict”	PICT	0x380A
filename extension = “mp3”	MP3	0x3009
filename extension = “wav”	WAV	0x3008
filename extension = “mpg” or “mpeg”	MPEG	0x300B
filename extension = “mov” or “qt”	Quicktime Movie	0x300D
filename extension = “xml”	XML file	0x300E
filename extension = “avi”	AVI	0x300A
filename extension = “asf”	ASF	0x300C
filename extension = “txt” or “text”	Text	0x3004
filename extension = “htm” or “html”	HTML	0x3005
Unknown Image File	Undefined	0x3800
Totally Unknown	Undefined	0x3000

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Annex C
(informative)

Operation flow example scenarios

C.1 Pull examples

C.1.1 Scenario 1a

The initiator requests all image objects from the responder, ignoring thumbnails, non-image objects and associations.

Step	Initiator action	Parameter1	Parameter2	Parameter3	Responder action
1	GetDeviceInfo	0x00000000	0x00000000	0x00000000	Send DeviceInfo dataset
2	OpenSession	SessionID	0x00000000	0x00000000	Create ObjectHandles, StorageIDs if necessary
3	GetObjectHandles	0xFFFFFFFF	0xFFFFFFFF	0x00000000	Send ObjectHandle array
4	GetObjectInfo	ObjHandle 1	0x00000000	0x00000000	Send ObjectInfo 1 dataset
5	Repeat step 4 for each ObjectHandle	ObjHandle *n*	0x00000000	0x00000000	Send ObjectInfo *n* dataset
6	GetObject	ObjHandle 1	0x00000000	0x00000000	Send Object 1 data
7	Repeat step 6 for each ObjectHandle	ObjHandle *n*	0x00000000	0x00000000	Send Object *n* data
8	Close session	0x00000000	0x00000000	0x00000000	None

C.1.2 Scenario 1b

This scenario is another way of accomplishing the same result as scenario 1a, but rather than retrieving all of the ObjectInfo data sets prior to obtaining any of the objects, it retrieves one ObjectInfo data set, then retrieves that data set's corresponding object, before proceeding to the next ObjectInfo data set. This allows earlier access to actual object data and may result in improved perceived performance for situations with large numbers of objects.

Step	Initiator action	Parameter1	Parameter2	Parameter3	Responder action
1	GetDeviceInfo	0x00000000	0x00000000	0x00000000	Send DeviceInfo dataset
2	OpenSession	SessionID	0x00000000	0x00000000	Create ObjectHandles, StorageIDs if necessary
3	GetObjectHandles	0xFFFFFFFF	0xFFFFFFFF	0x00000000	Send ObjectHandle array
4	GetObjectInfo	ObjHandle 1	0x00000000	0x00000000	Send ObjectInfo 1 dataset
5	GetObject	ObjHandle 1	0x00000000	0x00000000	Send Object 1 data
6	Repeat step 4 for each ObjectHandle	ObjHandle *n*	0x00000000	0x00000000	Send ObjectInfo *n* dataset
7	Repeat step 5 for each ObjectHandle	ObjHandle *n*	0x00000000	0x00000000	Send Object *n* data
8	Close session	0x00000000	0x00000000	0x00000000	None

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

C.1.3 Scenario 2

The initiator requests all objects from responder, including non-image objects and associations.

Step	Initiator action	Parameter1	Parameter2	Parameter3	Responder action
1	GetDeviceInfo	0x00000000	0x00000000	0x00000000	Send DeviceInfo dataset
2	OpenSession	SessionID	0x00000000	0x00000000	Create ObjectHandles, StorageIDs if necessary
3	GetObjectHandles	0xFFFFFFFF	0x00000000	0x00000000	Send ObjectHandle array
4	GetObjectInfo	ObjHandle 1	0x00000000	0x00000000	Send ObjectInfo 1 dataset
5	Repeat step 4 for each ObjectHandle	ObjHandle *n*	0x00000000	0x00000000	send ObjectInfo *n* dataset
6	GetObject	ObjHandle 1	0x00000000	0x00000000	Send Object 1 data
7	Repeat step 6 for each ObjectHandle	ObjHandle *n*	0x00000000	0x00000000	Send Object *n* data
8	Close session	0x00000000	0x00000000	0x00000000	None

C.1.4 Scenario 3

The initiator requests all thumbnails from the responder, ignoring non-images and associations. The initiator then requests a subset of the image objects.

Step	Initiator action	Parameter1	Parameter2	Parameter3	Responder action
1	GetDeviceInfo	0x00000000	0x00000000	0x00000000	Send DeviceInfo dataset
2	OpenSession	SessionID	0x00000000	0x00000000	Create ObjectHandles, StorageIDs if necessary
3	GetObjectHandles	0xFFFFFFFF	0xFFFFFFFF	0x00000000	Send ObjectHandle array
4	GetObjectInfo	ObjHandle 1	0x00000000	0x00000000	Send ObjectInfo 1 dataset
5	Repeat step 4 for each ObjectHandle	ObjHandle *n*	0x00000000	0x00000000	Send ObjectInfo *n* dataset
6	GetThumb	ObjHandle 1	0x00000000	0x00000000	Send Thumb 1 data
7	Repeat step 6 for each ObjectHandle	ObjHandle *n*	0x00000000	0x00000000	Send Thumb *n* data
8	GetObject	ObjHandle *a*	0x00000000	0x00000000	Send Object *a* data
9	Repeat step 8 for each ObjectHandle selected	ObjHandle *m*	0x00000000	0x00000000	Send Object *m* data
10	Close session	0x00000000	0x00000000	0x00000000	None

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

C.1.5 Scenario 4

The initiator examines the list of stores on the responder. The initiator chooses one store and discovers all objects in the root of the hierarchical filesystem of that store. The initiator then retrieves all thumbnails for any image objects discovered there.

Step	Initiator action	Parameter1	Parameter2	Parameter3	Responder action
1	GetDeviceInfo	0x00000000	0x00000000	0x00000000	Send DeviceInfo dataset
2	OpenSession	SessionID	0x00000000	0x00000000	Create ObjectHandles, StorageIDs if necessary
3	GetStorageIDs	0x00000000	0x00000000	0x00000000	Send StorageID array
4	GetStorageInfo	StorageID 1	0x00000000	0x00000000	Send StorageInfo 1
5	Repeat step 4 for each ObjectHandle in array returned in step 4	StorageID *n*	0x00000000	0x00000000	Send StorageInfo *n*
6	GetObjectHandles	StorageID of selected store	0x00000000	0xFFFFFFFF	Send ObjectHandle array for objects in root of selected store
7	GetObjectInfo	ObjHandle 1	0x00000000	0x00000000	Send ObjectInfo 1 dataset
8	Repeat step 7 for each ObjectHandle in array returned in step 6	ObjHandle *n*	0x00000000	0x00000000	Send ObjectInfo *n* dataset
9	GetThumb	ObjHandle of first image object	0x00000000	0x00000000	Send Thumb 1 data
10	Repeat step 9 for each ObjectHandle in array returned in step 6 that represents an image	ObjHandle *n*	0x00000000	0x00000000	Send Thumb *n* data
11	Close session	0x00000000	0x00000000	0x00000000	None

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

C.2 Push examples

C.2.1 Scenario 1

The initiator pushes all objects to the responder, allowing the responder to determine where to place the objects it is receiving.

Step	Initiator action	Parameter1	Parameter2	Parameter3	Responder action
1	GetDeviceInfo	0x00000000	0x00000000	0x00000000	Send DeviceInfo dataset
2	OpenSession	SessionID	0x00000000	0x00000000	Create ObjectHandles, StorageIDs if necessary
3	SendObjectInfo	0x00000000	0x00000000	0x00000000	Allocate memory, assign new ObjHandle, return StorageID, parent ObjHandle, ObjHandle
4	SendObject	0x00000000	0x00000000	0x00000000	Write data to store, invalidate SendObjInfo
5	Repeat steps 3 and 4 for each object to send in a top-down fashion	—	—	—	—
6	Close session	0x00000000	0x00000000	0x00000000	None

C.2.2 Scenario 2

The initiator pushes all objects to the responder, specifying the intended location on the responder for each.

Step	Initiator action	Parameter1	Parameter2	Parameter3	Responder action
1	GetDeviceInfo	0x00000000	0x00000000	0x00000000	Send DeviceInfo dataset
2	OpenSession	SessionID	0x00000000	0x00000000	Create ObjectHandles, StorageIDs if necessary
3	SendObjectInfo	Responder target StorageID	Responder target parent's ObjectHandle	0x00000000	Allocate memory, assign new ObjHandle, return StorageID, parent ObjHandle, ObjHandle
4	SendObject	0x00000000	0x00000000	0x00000000	Write data to store, invalidate ObjInfo
5	Repeat steps 3 and 4 for each object to send in a breadth-wise fashion	—	—	—	—
6	Close session	0x00000000	0x00000000	0x00000000	None

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Annex D
(informative)

Filesystem implementation examples

D.1 Scope

This annex shows examples of how to implement support for filesystems.

D.2 ObjectHandle assignment

Regardless of the filesystem used for a particular storage device, files stored in the filesystem may be assigned any handle. The simplest method for ObjectHandle creation is to traverse the filesystem hierarchy in a breadth-first or depth-first fashion, assigning ObjectHandles to each directory and file using consecutively numbered values starting with 0x00000001.

Objects may be created to represent associations that do not exist as folders in the filesystem, according to the convention used by the particular device; e.g. this type of association might be determined from the naming convention used for the individual files that are part of the association. An example would be a store that contained only three files that were part of a burst association as indicated by the filenames burst001.jpg, burst002.jpg and burst003.jpg. In this case, four ObjectHandles would be assigned; one for each of the images in the association (e.g. 0x00000001, 0x00000002, 0x00000003), and an extra handle to represent the virtual association that is the burst relationship (e.g. 0x00000004). The ObjectInfo data sets returned for each of the image objects should contain a ParentObject field with a value equal to the ObjectHandle of the virtual folder (i.e. 0x00000004). No GetObject would be necessary for the object with handle 0x00000004 because associations may be fully described and reproduced in the most appropriate form by examining the ObjectInfo data set of the association.

D.3 DCF filesystem association example

Associations may be used to represent filesystems. The following is an example showing a typical filesystem on a DSPD that conforms with DCF. This example shows examples of how a DPOF file would be handled, as well as generic folder associations, a burst association and ancillary data associations, including both an audio file associated with a single Exif, as well as a text file associated with an entire burst sequence.

DCF filesystem directory example:

\MISC\AUTPRINT.MRK // DPOF text file

\DCIM\100MODEL\ // DSPD-model-specific folder

\DCIM\100MODEL\DCP_0001.JPG // Exif Image #1

\DCIM\100MODEL\DCP_0002.JPG // Exif image #2

\DCIM\100MODEL\DCP_0002.WAV // WAV file assoc’ed with Exif #2

\DCIM\100MODEL\B01_0003.JPG // Burst Set #1 (1/3) Exif #3

\DCIM\100MODEL\B01_0004.JPG // Burst Set #1 (2/3) Exif #4

\DCIM\100MODEL\B01_0005.JPG // Burst Set #1 (3/3) Exif #5

\DCIM\100MODEL\B01_0003.TXT // Text file associated w/ burst seq

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

See Table D.1.

Table D.1 — DCF filesystem example

Object handle	Object description	ObjFormat code	Parent object	Seq. number
0x00000001	\MISC folder	0x3001	0x00000000	0x00000000
0x00000002	\MISC\AUTPRINT.MRK	0x3006	0x00000001	0x00000000
0x00000003	\DCIM folder	0x3001	0x00000000	0x00000000
0x00000004	\DCIM\100MODEL folder	0x3001	0x00000003	0x00000000
0x00000005	\DCIM\100MODEL\DCP_0001.JPG	0x3801	0x00000004	0x00000000
0x00000006	Ancillary Data Assoc. for Exif#2/WAV	0x3001	0x00000004	0x00000000
0x00000007	\DCIM\100MODEL\DCP_0002.JPG	0x3801	0x00000006	0x00000000
0x00000008	\DCIM\100MODEL\DCP_0002.WAV	0x3008	0x00000006	0x00000000
0x00000009	Burst Association for Burst Seq. #1	0x3001	0x00000004	0x00000000
0x0000000A	\DCIM\100MODEL\B01_0003.JPG	0x3801	0x00000009	0x00000001
0x0000000B	\DCIM\100MODEL\B01_0004.JPG	0x3801	0x00000009	0x00000002
0x0000000C	\DCIM\100MODEL\B01_0005.JPG	0x3801	0x00000009	0x00000003
0x0000000D	\DCIM\100MODEL\B01_0003.TXT	0x3004	0x00000009	0x00000000

D.4 GetFilesystemManifest example (PTP v1.1)

Continuing with the example in the previous clause, this example shows how the optional GetFilesystemManifest operation would return all of the fast relevant filesystem information in one contiguous data set array.

See Table D.2.

Table D.2 — GetFilesystemManifest dataphase (PTP v1.1) — Optional

Dataset field	**Field size** (bytes)	**Field value**
NumberDatasets	8	13 (0x00000000-0000000D)
ObjectHandle	4	0x00000001
StorageID	4	0x00010001
ObjectFormat	2	0x3001
ProtectionStatus	2	0x0000
ObjectCompressedSize	8	0x00000000-00000000
ParentObject	4	0x00000000
AssociationType	2	0x0001
AssociationDesc	4	0x00000000
SequenceNumber	4	0x00000000
Filename	Variable	“MISC”
ObjectHandle	4	0x00000002
StorageID	4	0x00010001
ObjectFormat	2	0x3006
ProtectionStatus	2	0x0000
ObjectCompressedSize	8	(size in bytes of autprint.mrk file)

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Table D.2 *(continued)*

Dataset field	**Field size** (bytes)	**Field value**
ParentObject	4	0x00000001
AssociationType	2	0x0000
AssociationDesc	4	0x00000000
SequenceNumber	4	0x00000000
Filename	Variable	“AUTPRINT.MRK”
… (10 more similar datasets)		
ObjectHandle	4	0x0000000D
StorageID	4	0x00010001
ObjectFormat	2	0x3004
ProtectionStatus	2	0x0000
ObjectCompressedSize	8	(size in bytes of B01_0003.TXT file)
ParentObject	4	0x00000009
AssociationType	2	0x0000
AssociationDesc	4	0x00000000
SequenceNumber	4	0x00000000
Filename	Variable	“B01_0003.TXT”

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Annex E
(informative)

Reference to OSI model

E.1 Open systems interconnection (OSI)

Parts of OSI have influenced Internet protocol development, but none more than the abstract model itself, documented in ISO 7498 and its various addenda. In this model, a networking system is divided into layers. Within each layer, one or more entities implement its functionality. Each entity interacts directly only with the layer immediately beneath it, and provides facilities for use by the layer above it. Protocols enable an entity in one host to interact with a corresponding entity at the same layer in a remote host. See Figure E.1.

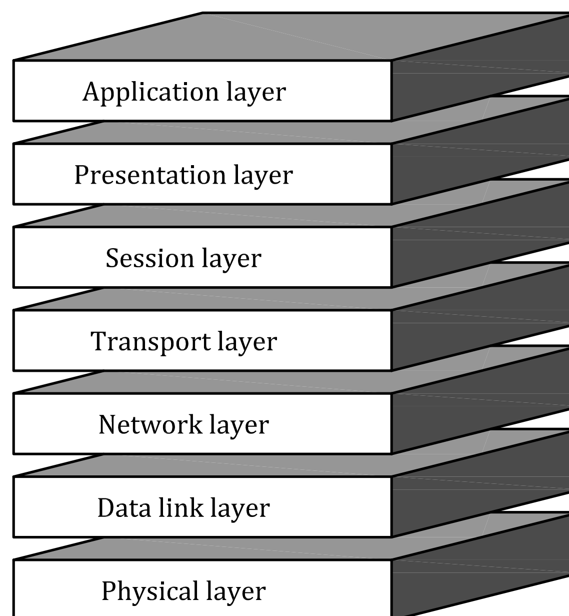

Figure E.1 — The seven layers of the OSI basic reference model

The **Physical layer** describes the physical properties of the various communications media, as well as the electrical properties and interpretation of the exchanged signals; e.g. this layer defines the size of Ethernet coaxial cable, the type of BNC connector used and the termination method.

The **Data link layer** describes the logical organization of data bits transmitted on a particular medium; e.g. this layer defines the framing, addressing and check summing of Ethernet packets.

The **Network layer** describes how a series of exchanges over various data links can deliver data between any two nodes in a network; e.g. this layer defines the addressing and routing structure of the Internet.

The **Transport layer** describes the quality and nature of the data delivery; e.g. this layer defines if and how retransmissions will be used to ensure data delivery.

The **Session layer** describes the organization of data sequences larger than the packets handled by lower layers; e.g. this layer describes how request and reply packets are paired in a remote procedure call.

The **Presentation layer** describes the syntax of data being transferred; e.g. this layer describes how floating point numbers can be exchanged between hosts with different maths formats.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

The **Application layer** describes how real work actually gets done; e.g. this layer implements filesystem operations.

E.2 Picture transfer protocol implemented over USB

Picture transfer protocol implemented over USB can be viewed as having a four-layer architecture. See Figure E.2.

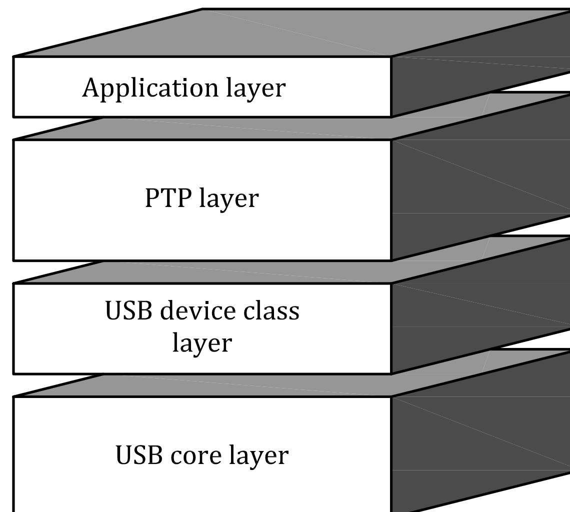

Figure E.2 — The four layers of PTP implemented over USB

The **USB core layer** describes the physical properties of the USB cable, its electrical properties and interpretation of the exchanged signals. This layer defines the connector used, the termination method and the signal levels. This layer has some specific properties that distinguish it from the generic OSI physical layer. USB utilizes a master/slave connection model typical of peripheral interfaces. OSI is usually applied to network interfaces that are based on peer-to-peer connection models. USB supports a framing protocol that enables time-based synchronous data transfers to occur. Network protocols do not support this directly. The OSI physical layer does not account for the fact that data packet exchange can occur in different ways with different behaviour and properties. USB defines different modes for exchanging data packets (isochronous, bulk, control and interrupt transfers – refer to the USB core specification). In addition to defining the data transfer or framing methods, the USB core layer specification also defines the format of the content of specific types of data objects called descriptors. These descriptors help describe the properties of the connected device, a characteristic of peripheral interfaces but not in a network interface. In general, the USB core layer maps to layers 1, 2, 3 and 4 of the OSI model.

The **USB device class layer** best corresponds to layer 5, the session layer in the OSI model, however, it also has functionality that corresponds to layer 4 in the OSI model. The USB device class layer defines the format of a larger data container that spans several data packets. It defines containers for the basic commands, data, responses and events that are utilized by PTP. The USB device class layer defines the protocol needed to recover from errors in the exchange of data containers. Refer to the USB still image capture device definition.

The **PTP layer** corresponds to layers 5 and 6 in the OSI model. PTP describes the specific data format or syntax of several objects encapsulated in the data packets. Where this is not true, the format of the object is defined in other standards. PTP describes the various codes needed for the operation of the application layer. The PTP layer defines descriptive objects (PTP data sets) that contain information that is, in places, somewhat redundant to the information in the descriptors found in the USB core layer. This situation arises whenever PTP is placed over a peripheral interface. USB and IEEE 1394 have this characteristic. Should PTP operate over Ethernet using only network protocols this would not be the case. When any redundancy occurs, the application layer should give priority to the information in the PTP data sets. Device implementers should take care to avoid any contradiction between any descriptors and data sets.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

The **Application layer** describes how real work actually gets done. This layer contains the firmware controlling the connected device or the host application, and corresponds to layer 7 of the OSI model.

E.3 Picture transfer protocol's generic architecture

Regardless of which transport is used to host picture transfer protocol (PTP), PTP will generally correspond to layers 5 and 6 of the OSI model. Generally, the lower layer must provide a reliable data transport, be able to provide a means of distinguishing the command data and response phases of the PTP protocol, and provide a means of supporting events. It may be necessary to define a layer above the core transport to encapsulate PTP. Such was the case with the USB, as the USB device class layer.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Annex F
(informative)

SendObject implementation example

F.1 Scope

This annex provides an example of how the SendObject and SendObjectInfo operations are implemented by both the initiator (in this case the sender or host computer) and the responder (in this case the receiver or PTP camera).

F.2 Preparations

Before initiating a SendObject sequence, the following conditions will have occurred:

a) a session has been opened;

b) the initiator has examined the DeviceInfo data set of the responder to ensure that the SendObjectInfo and SendObject operations are in the OperationsSupported field.

F.3 SendObject sequence of PTP operations and events

a) The initiator invokes the SendObjectInfo operation.

b) If the object is not an association and the SendObjectInfo operation was successful, the initiator invokes the SendObject operation.

c) (Multisession implementations only). The responder initiates an ObjectAdded event to any OTHER sessions that are open. This event would NOT be sent to the session for which the SendObject is occurring, and therefore would not occur for single-session transports and/or devices (e.g. the currently defined USB implementation).

d) The initiator optionally invokes the GetObjectInfo operation on the sent object if it desires to know exactly how the object was stored by the responder.

F.4 SendObject detailed behaviour

a) The initiator issues a SendObjectInfo (hereafter referred to as SOI) operation to the responder. The dataphase of this operation includes the sending of an ObjectInfo Data set (hereafter referred to as OID) that is required by this International Standard to possess at least the following fields:

   1) ObjectFormatCode;

   2) ObjectCompressedSize;

   3) AssociationType (only if the object is an association/folder).

b) The OID is stored by the receiver, overwriting any OID that may be already there, and ensuring at this point that the OID buffer is considered not yet valid, as it has not yet been examined.

c) The responder checks for the following potential error conditions:

   1) that the operation is supported; if not, the responder returns Operation_Not_Supported;

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

2) the first parameter (destination Storage ID); if this parameter is non-zero:

i) checks to see that it is a valid store; if not, it returns Invalid_StorageID;

ii) if it is possible for the indicated store not to be available (e.g. it is removable), the responder ensures that it is available; if not, Store_Not_Available is returned;

iii) ensures that the indicated store is writable; if not, the responder returns Store_Read_Only;

3) the second parameter (parent ObjectHandle); if this parameter is non-zero:

i) if the responder does not support this functionality, return Specification_of_Destination_Unsupported; this indicates to the sender that it should refrain from attempting to specify this parameter in the future, at least for objects of this particular ObjectFormatCode;

ii) if the ParentObject referred to is not an association, the responder returns Invalid_ParentObject;

4) the OID for the following:

i) general validity; if malformed, the responder returns Invalid_Data set;

ii) presence of all required fields:

I) ObjectFormatCode;

II) ObjectCompressedSize;

III) AssociationType (only if the object is an association/folder).

If a required field is not present, the responder returns Invalid_Data set. This implies that all other fields in the OID are optional, including the Filename and StorageID fields. Optional means that the sending device may or may not populate these fields, not that the receiver may or may not pay attention to them. Therefore, the receiver must be able to handle situations where any optional fields are not specified.

5) The responder determines whether there is a suitable destination with enough space by examining the ObjectCompressedSize field of the OID:

i) If the first parameter was non-zero, the responder ensures that there is enough space in the indicated store; if not, the responder returns Store_Full.

ii) If the first parameter was zero, the responder determines where to put the incoming object. Each acceptable, available, writable store is checked in order to determine if there is enough space somewhere. For devices with multiple stores, any algorithm may be used to determine which store is the best (e.g. device-stored user preference, first store examined that has enough space, last store used, store with the most available space, selecting store based on ObjectFormatCode considerations, etc.). If there is no suitable store for reasons other than space considerations, the responder returns Store_Not_Available. If there is a suitable store, but none with enough space, the responder returns Store_Full.

d) If no error conditions have occurred, the responder now considers the OID that it received from the initiator as being valid. Therefore, when the SendObject operation is issued later, the responder will know that it should associate that operation with the OID in its buffer.

e) The responder then allocates the space indicated in the ObjectCompressedSize of the OID, in whatever manner is appropriate, in order to ensure that the available space does not disappear before storing of the object is complete, due to any unrelated manual captures or other sessions for multisession-capable transports/devices. If the SOI operation is successful, the responder retains the OID in its one-and-only buffer used for this purpose, and returns a successful ResponseCode to the sender (i.e. OK), along with the StorageID, ParentObjectHandle and ObjectHandle that the receiver will assign to the new object as response parameters. If the object being sent is of type

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Association, then no SendObject operation is required, as the association object is fully described by the OID. In this case, proceed to step g).

f) If the object is not an association, the sender invokes a SendObject operation, and during this operation's dataphase, the actual binary object is sent from the initiator (i.e. sender) to the responder (i.e. receiver). Most error cases should have been caught prior to the object being sent, as part of the proceeding SOI operation. If there is no valid OID in the buffer, or the OID that is in the buffer does not match the object (i.e. ObjectCompressedSize) then the response No_Valid_ObjectInfo is returned. If this error response occurs, and there was an OID in the receiver buffer but its size does not match that of the object just sent, then the object is discarded and the OID held in the responder's buffer is invalidated. Transmission of the binary object may be optionally avoided by cancelling the dataphase as described by the transport implementation.

g) The responder creates the object in the appropriate location. If the responder uses a filesystem for storage, it may check the filename field of the OID and use that filename directly if it is available and appropriate, or change the name to something that is suitable. The receiver then invalidates its OID buffer. The following issues may arise if the responder attempts to use the supplied filename field in the OID:

   1) the filename may be longer than the receiver can handle (e.g. 8.3 vs. long filename considerations);

   2) the filename may conflict with a filename already at the intended/determined destination;

   3) the filename may be in a language that the receiver's internal filesystem does not support (e.g. it does not support UNICODE).

   In any of the above cases, the responder should not attempt to fail the operation, as the primary mechanism for image referencing in PTP is by ObjectHandle and not filename. In the above filename error cases, the responder should create its own filename for internal storage of the newly received object. The following conditions also apply:

   4) if the sender suggests a filename, the responder may wish to use it as the basis for creating its internal filename, through adding a prefix, suffix or some name-mangling algorithm, or it may wish to generate a completely new filename, perhaps using the same mechanism that is used to name any newly captured objects;

   5) the responder must be able to handle the case where no filename is suggested by the sender in the OID;

   6) the initiator should never use any filesystem extension (e.g. .jpg) that may exist within a suggested filename without first checking it against the ObjectFormatCode supplied in the OID received as part of the SOI operation; for systems with traditional filesystems, if the responder needs to specify a filename extension for its own internal storage, it should generate the extension from the ObjectFormatCode.

h) (Multisession only) The responder initiates an ObjectAdded event to all OTHER sessions that are currently open. This event is NOT sent to the session for which the SendObject is occurring, and therefore will not occur for single-session transports and/or devices (and therefore does not occur for the current USB implementation).

i) If the initiator views the responder in a filesystem-like view, and it needs to update its view of the responder to show the newly sent object, then it should issue a GetObjectInfo on the newly placed object, using the ObjectHandle returned as the third response parameter of the previously invoked SOI operation. This is because the responder may or may not have used the suggested filename.

© ISO 2013 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

Bibliography

[1] CIPA DC-009-2010, *Design rule for Camera File system* : DCF Version 2.0 (Edition 2010) available from http://www.cipa.jp/english/hyoujunka/kikaku/pdf/DC-009-2010_E.pdf

[2] *USB Specification,* available from http://www.usb.org/developers/docs/

[3] *Universal Serial Bus Still Image Capture Device Definition, Version 1.0*, USB Implementer's Forum, 11 July, 2000, available from http://www.usb.org/developers/devclass_docs/usb_still_img10.pdf

[4] IEEE 1394-1995, *IEEE Standard for a High Performance Serial Bus*, IEEE 1394 information available via world-wide web at: http://www.1394ta.org

[5] IIDC 1394-based Digital Camera specification, Version 1.30, July 25, 2000. Available from http://www.cs.unc.edu/Research/stc/FAQs/1394Firewire/DCAM_Spec_V1_30.pdf

[6] PTP-on-SBP (*Picture Transfer Protocol on Serial Bus Protocol*), IEEE 1394 Trade Association

[7] *FlashPix Specification* is available from http://www.graphcomp.com/info/specs/livepicture/fpx.pdf

[8] IrDA information available from http://www.irda.org

[9] Melville J. et al.*An application programmer's interface for digital cameras*, Proc. IS&T's 49th Annual Conference, pp. 282-285, May 1996

[10] *PNG Information* is available from http://www.w3.org/TR/PNG/

[11] PICT specification *Inside Macintosh: Imaging with QuickDraw*, ISBN: 020163242X, Addison Wesley Publishing Company, 1994; PDF at https://developer.apple.com/legacy/library/documentation/mac/pdf/ImagingWithQuickDraw.pdf

[12] ISO 7498:1984[2)], *Information processing systems — Open Systems Interconnection — Basic Reference Model*

[13] *Transmission Control Protocol*, Internet Engineering Task Force (IETF) RFC 793, by Information Sciences Institute, University of Southern California, September 1981; http://ietf.org/rfc/rfc793.txt

[14] CIPA DC-005-2005, "Picture Transfer Protocol" over TCP/IP networks (PTP-IP). Information available from http://www.cipa.jp/ptp-ip/index_e.html

[15] ISO 12234-1, *Electronic still-picture imaging — Removable memory — Part 1: Basic removable-memory model*

2) Replaced by ISO 7498-1.

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST

ICS 37.040.99

Price based on 115 pages

© ISO 2013 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=sharabiani, shahramfs
Not for Resale, 11/29/2013 00:19:37 MST