Media Transfer Protocol

Revision 1.0

December 5th, 2007

Please send comments via electronic mail to mtp-chair@usb.org

Table of Contents

1 Introduction

Media Transfer Protocol, or MTP, is a protocol designed for content exchange with and command and control of transient storage devices. It was been developed as an extension to PTP, or Picture Transfer Protocol, and is targeted primarily at Digital Still Cameras, Portable Media Players and Cellular phones.

1.1 Purpose

The primary purpose of this protocol is to facilitate communication between media devices that have transient connectivity and significant storage capacity. This includes the exchange of binary objects and the enumeration of the contents of that connected device.

The secondary purpose of this protocol is to enable command and control of the connected device. This includes the remote invocation of device functionality, monitoring of device-initiated events, and the reading and setting of device properties.

1.2 MTP Device Model

MTP devices may be loosely defined as devices with storage that consume or produce media objects in a binary format, which have intermittent connections with other media devices, and which fulfill their primary purpose while not connected to another device.

Devices generally act primarily as either media consumers or media producers, although this line is becoming increasingly blurred. Some examples of common portable media devices are: digital cameras (both still and video), portable audio players, and cellular phones.

1.3 MTP Object Model

The term "media" in "Media Transfer Protocol" is used to identify any binary data, and is not restricted to audio/video formats to which it is commonly applied. Some examples of non-audio/video objects include contacts, programs, scheduled events and text files.

Media objects are required to be represented as atomic binary objects during transfer, but are not required to be stored in the same format or structure on the device. Objects may be created on-demand, as long as they are accurately represented during content enumeration.

MTP objects consist of not only the binary content of the file, but also the descriptive metadata and references. MTP is designed such that objects can be recognized through the mechanisms provided in the protocol without requiring an understanding of the binary format of the file itself.

The combination of the binary file, its descriptive metadata and any intra-object references together is referred to in MTP as an object.

1.4 Scope

This specification is intended to define the USB implementation of MTP in a way that is agnostic to both device type and OS. Certain operating systems or device classes may require a particular subset of MTP to enable their minimal scenarios; it is strongly suggested that implementers investigate the intended usage scenarios to determine if any such requirements exist.

1.5 PTP Compatibility

This protocol is implemented as an extension of the existing Picture Transfer Protocol, as defined by the ISO 15740 specification (http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=37445&ICS1=37&ICS2=40&ICS3=99). MTP is intended to co-exist with and not overlap PTP functionality, and it is hoped that devices will be developed to comply fully with the PTP specification where possible to leverage the existing base of PTP-enabled devices and applications.

MTP has been implemented using the defined vendor extensibility mechanism of PTP using the MTP Vendor Extension ID. By implementing MTP in this way, compatibility of devices with PTP is preserved. However, in rare cases, optional modifications to the core protocol have been included to enhance the functionality of connected devices. These are clearly identified in this specification, and it is the responsibility of implementers of this protocol to determine whether PTP-compatibility is desirable, and if so, to implement this protocol in such a manner as to be compatible with PTP.

The name "Media Transfer Protocol" is used to refer to the combination of the core PTP specification and the extension set provided by the USB-IF in this specification.

2 Transport Requirements

MTP is intended to be transport-agnostic, that is, it is intended to function over multiple underlying transports. However, this specification deals specifically with the USB implementation of MTP – which assumes certain qualities in the USB transport in order to function effectively.

2.1 Disconnection Events

An underlying transport should notify the application or service that is implementing MTP initiator or responder functionality when a connected MTP device is disconnected or otherwise rendered inaccessible at the transport level.

2.2 Error-Free Data Transmission

An underlying transport should guarantee error-free data transmission. This may be provided by the transport itself, using error-correction codes or packet validation, or may be ensured artificially on an inconsistent transport by the transport-specific implementation definition.

2.3 Asynchronous Events

MTP devices are required to notify any connected devices immediately about any changes in device status, device properties, object addition/deletion/modification or storage status modifications, and so on. In order to provide that support, the underlying transport must enable events to be communicated asynchronously with operations, responses or data transfers.

2.4 Device Discovery and Enumeration

MTP does not attempt to define how devices are discovered or identified as supporting MTP. This should be defined in a manner consistent with the underlying transport, and may be performed in more than one way for a given transport.

2.5 Security and Authentication

MTP does not include any functionality for user authentication or data security. Any sort of device validation or protection of data while in transit should be implemented in a transport-specific manner.

2.6 Transport Independence

MTP was fundamentally defined as a transport independent protocol, while this specification serves to define a standard USB implementation only. As a result, while this specification shall be considered the definitive protocol reference, nothing in this specification shall preclude the core protocol from operating over other transports. These may include, but are not limited to, TCP/IP, Bluetooth, serial, or any other yet to be defined transport mechanism.

2.7 MTP Device Enumeration

MTP device enumeration over USB requires no specific or proprietary USB string descriptors. Some implementations prior to the publication of this specification may require proprietary enumeration techniques; those are not specifically covered in this document.

3 Normative Reference

Within the context of MTP, certain terms and conventions are used extensively to provide a basic foundation of functionality in the protocol. These conventions are described here.

MTP does not require that the device employ these conventions or representations in the internal working of a device, only that they are adhered to in the implementation of the over-the-wire protocol.

3.1 Data Formatting

PTP defines a convention for encoding datatypes and datasets, and as a PTP extension set the data structures used in MTP are identical to those used in PTP. It is repeated here for convenience, but the PTP specification shall (as always) be considered the definitive source.

3.1.1 Multi-byte Data

The standard format for multi-byte data in this specification is big-endian. That is, the bits within a byte will be read such that the most significant byte is read first. The actual multi-byte data sent over the transport may not necessarily adhere to this same format, and the actual multi-byte data used on the devices may also use a different multi-byte format. The big-endian convention only applies within this document, except where otherwise stated.

3.1.2 Bit Field Format

When bit fields are defined in this format, the least significant bit is at the zero position. When the bit field is represented in the specification, this is the right-most position. For example, the most significant bit of a 32-bit integer (UINT32) will be at the 31st position, while the least significant bit will be at the 0-th position.

3.2 Simple Types

All non-opaque data in MTP consists of either an atomic value of a simple type, or an array of atomic values of a simple type. The set of simple atomic types used in MTP is described here as a common foundation for data representation.

Specifically, any time data is passed as a parameter to an operation, response or event, it must take one of the following forms.

3.2.1 Simple Type Summary

Data Type 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	Undefined	Reserved (PTP)

3.2.2 Arrays

Arrays are defined in PTP (and thus MTP) as a concatenation of the same fixed-length type. MTP does not define an array of strings. The size of each element is identified by the simple type that is contained in the array (see 3.2.2.1). Arrays in MTP start with an unsigned 32-bit integer that identifies the number of elements to follow, followed by a concatenation of repeated instances of the simple type identified by the array's datatype code. For the purposes of this specification, arrays are considered to be zero-based.

An empty array is represented by a single 32-bit integer containing a value of 0x00000000.

3.2.2.1 Array Definition

Field	Size (bytes)	Format
NumElements	4	UINT32
ArrayEntry[0**]**	Element Size	Special
ArrayEntry[1**]**	Element Size	Special
…	…	…
ArrayEntry[*NumElements-1***]**	Element Size	Special

3.2.3 Strings

Strings in PTP (and thus MTP) consist of standard 2-byte Unicode characters as defined by ISO 10646. Strings begin with a single, 8-bit unsigned integer that identifies the number of characters to follow (**not** bytes). An empty string is represented by a single 8-bit integer containing a value of 0x00. A non-empty string is represented by the count byte, a sequence of Unicode characters, and a terminating Unicode L'\0' character ("null"). Strings are limited to 255 characters, including the terminating null character.

It should be noted that strings with embedded nulls are not permitted.

Examples:
The string L"" is represented as the single byte 0x00.
The string L"A" is represented as the five-byte sequence 0x02 0x41 0x00 0x00 0x00.

3.2.3.1 String Definition

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

3.2.4 Decimal Types

MTP does not include decimal values as a data type. They may be represented using the string datatype where required, or the unit of measurement can be subdivided to allow a particular level of precision (for example, measure in thousandths instead of having decimals to three places).

3.2.5 DateTime

DateTime strings follow a compatible subset of the definition found in ISO 8601, and take the form of a Unicode string formatted as: "YYYYMMDDThhmmss.s". In this representation, YYYY shall be replaced by the year, MM replaced by the month (01-12), DD replaced by the day (01-31), T is a constant character 'T' delimiting time from date, hh is replaced by the hour (00-23), mm is replaced by the minute (00-59), and ss by the second (00-59). The ".s" is optional, and represents tenths of a second.

This string can optionally be appended with a constant character "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.

Leap seconds are not used in MTP.

The following regular expression accurately defines DateTime strings:
[0-9]{4}(0[1-9]|1[0-2])(0[1-9]|[12][0-9]|3[01])T([01][0-9]|2[0-3])([0-5][0-9])([0-5][0-9])(\.[0-9])?(Z|[+-]([01][0-9]|2[0-3])([0-5][0-9]))?

3.3 Datacodes

In MTP, all protocol traffic is binary and of fixed length. In order to enable this, all operations, responses, events, object formats, and properties are represented by assigned 16-bit datacodes.

Datacodes are assigned from a range which is partitioned by origin (PTP, MTP or Vendor Extension) as well as function. Though this assignment is explicitly partitioned, the location of a datacode in a range in the table below shall not be used to interpret the datacode; rather, all datacodes shall be individually recognized by their specific value.

As the MTP vendor-extension set occupies the vendor-extension datacode ranges of PTP, the MTP ranges have been further subdivided to allow vendor extensions to MTP.

3.3.1 Datacode Summary

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bits 9-0	Values	Datacode type
0	0	0	0	Any	Any	Any	0..0xFFF	Undefined
0	0	0	1	Any	Any	Any	0x1000.. 0x1FFF	PTP Operation Code
0	0	1	0	Any	Any	Any	0x2000.. 0x2FFF	PTP Response Code
0	0	1	1	Any	Any	Any	0x3000.. 0x3FFF	PTP Object Format Code
0	1	0	0	Any	Any	Any	0x4000.. 0x4FFF	PTP Event Code
0	1	0	1	Any	Any	Any	0x5000.. 0x5FFF	PTP Device Prop Code
0	1	1	0	Any	Any	Any	0x6000.. 0x6FFF	Reserved (PTP)
0	1	1	1	Any	Any	Any	0x7000.. 0x7FFF	Reserved (PTP)

1	0	0	0	Any	Any	Any	0x8000.. 0x8FFF	Undefined
1	0	0	1	0	Any	Any	0x9000.. 0x97FF	Vendor Extension Operation Code
1	0	0	1	1	Any	Any	0x9800.. 0x9FFF	MTP Operation Code
1	0	1	0	0	Any	Any	0xA000.. 0xA7FF	Vendor Extension Response Code
1	0	1	0	1	Any	Any	0xA800.. 0xAFFF	MTP Response Code
1	0	1	1	0	Any	Any	0xB000.. 0xB7FF	Vendor Extension Object Format Code
1	0	1	1	1	Any	Any	0xB800.. 0xBFFF	MTP Object Format Code
1	1	0	0	0	Any	Any	0xC000.. 0xC7FF	Vendor Extension Event Code
1	1	0	0	1	Any	Any	0xC800.. 0xC8FF	MTP Event Code
1	1	0	1	0	0	Any	0xD000.. 0xD3FF	Vendor Extension Device Prop Code
1	1	0	1	0	1	Any	0xD400.. 0xD7FF	MTP Device Prop Code
1	1	0	1	1	0	Any	0xD800.. 0xDBFF	Vendor Extension Object Prop Code
1	1	0	1	1	1	Any	0xDC00.. 0xDFFF	MTP Object Prop Code
1	1	1	0	Any	Any	Any	0xE000.. 0xEFFF	Reserved (PTP)
1	1	1	1	Any	Any	Any	0xF000.. 0xFFFF	Reserved (PTP)

Individual datacode types are explained in the appropriate sections of this document.

3.4 Object Handles

Object handles are 32-bit identifiers that provide a device- and session-unique consistent reference to a logical object on a device. All handles are represented using the UINT32 simple type. There is no special meaning attached to the value of object handles; they may be chosen in any manner that facilitates device implementation.

Object handles are used in MTP transactions to reference a logical object on the device, but do not necessarily reference actual data constructs on the device. As identified in

section 1.3, objects may be exposed through MTP that will be created on-demand. Object handles are only persistent within an MTP session; once a session has been re-opened, all previous values shall be assumed to be invalid, and the contents of the Responder must be re-enumerated if object handles are needed.

The values "0xFFFFFFFF" and "0x00000000" have special meaning and shall not be assigned to objects. The meanings of those values are context-specific.

3.4.1 Assigning Object Handles

Object Handles are assigned by the responder. They must be globally unique across all storages on the device.

Object Handles are only defined within an open session. If the Initiator has not yet opened a session, object handles do not have any meaning and cannot be used. When a session is closed, either intentionally by the initiator or as a result of an error or USB interruption, all Object Handles are invalidated and must be re-acquired by the Initiator. Object Handles do **not** persist between sessions.

If an object is deleted in a session, the Responder shall not re-use the deleted object's Object Handle in the same session.

3.5 Object Formats

As MTP is file-system-agnostic, the usual method of overloading file extensions to determine file type is not a reliable method of identifying device contents. In many cases, objects on a device may have no qualified filename, or may not even exist until requested for transfer by the Initiator.

Instead, object formats are identified using predefined **ObjectFormat** datacodes.

3.5.1 Object Format Versions

The version information for a particular object format shall be contained in the object itself in a format-specific way, and declaring support for an object format type implies that the Responder is able to parse the data object to determine the appropriate version, and able to decode the data contained within.

If an object format has multiple versions all identified by the same **ObjectFormat** datacode, then any device that indicates support for that **ObjectFormat** shall be able to interpret and consume any version of that format. If an object format defined by this specification is not self versioning, different **ObjectFormat** types are assigned for each version. Vendor-defined **ObjectFormat** types shall follow this convention.

3.6 Associations

Associations are used in MTP to describe hierarchical file systems on devices. Responders and Initiators that are based on the PTP standard also use Associations to provide a limited method of associating related image and data objects.

The use of Associations has been superseded in MTP by the concept of object references. Refer to section 5.3.3 for more information about object references.

Associations other than type 0x1 (hierarchical) aren't used in MTP, and should not be unless a PTP device is being developed. In that case, the associations rules should be followed from the PTP specification and the object references references from this specification.

3.6.1 Association Types

Associations represent various collections of objects. The kind of collection is conveyed by the **Association Type** field in the **ObjectInfo** dataset, and is also exposed in the **Association Type** object property.

The **Association Description** field, also contained in the **ObjectInfo** dataset, provides additional information to further qualify the Association type. The meaning of the **Association Description** field varies depending on the **Association Type**.

3.6.2 Association Type Summary

Association code	**Association type**	**AssociationDesc interpretation**
0x0000	Undefined	Undefined
0x0001	Generic Folder	Unused by PTP; used by MTP to indicate type of folder
0x0002	Album	Reserved
0x0003	Time Sequence	Default Playback Delta
0x0004	Horizontal Panoramic	Unused
0x0005	Vertical Panoramic	Unused
0x0006	2D Panoramic	Images per row
0x0007	Ancillary Data	Undefined
All other values with bit 15 set to 0	Reserved	Unused
All other values with bit 15 set to 1 and bit 14 set to 0	Vendor-defined	Undefined
All other values with bit 15 set to 1 and bit 14 set to 1	MTP	Undefined

3.6.2.1 Generic Folder

Association objects with this Association Type represent hierarchical folders rooted on a particular storage. This provides a mechanism of exposing a file hierarchy on the device without relying on any path or naming conventions specific to a particular file system.

The AssociationDesc field of a Generic Folder Association may contain either 0x00000000 or 0x00000001. If it contains a value of 0x00000001, this indicates that it is a bi-directionally linked folder, and must have Object References to each object "contained" by this association (each object which contains this Association's ObjectHandle in the ParentID field of its ObjectInfo dataset).

Note that a PTP Initiator or Responder only has a defined response if a value of 0x00000000 is used in the AssociationDesc field.

3.6.2.2 Album

This Association Type is PTP-specific. For more information, please refer to the documents referenced in "USB Still Image Capture Device Definition – July 2000".

3.6.2.3 Time Sequence

This Association Type is PTP-specific. For more information, please refer to the documents referenced in "USB Still Image Capture Device Definition – July 2000".

3.6.2.4 Horizontal Panoramic

This Association Type is PTP-specific. For more information, please refer to the documents referenced in "USB Still Image Capture Device Definition – July 2000".

3.6.2.5 Vertical Panoramic

This Association Type is PTP-specific. For more information, please refer to the documents referenced in "USB Still Image Capture Device Definition – July 2000".

3.6.2.6 2D Panoramic

This Association Type is PTP-specific. For more information, please refer the documents referenced in "USB Still Image Capture Device Definition – July 2000".

3.6.2.7 Ancillary Data

This Association Type is PTP-specific. For more information, please refer the documents referenced in "USB Still Image Capture Device Definition – July 2000".

3.6.3 Associations as File System Folders

The primary usage of Associations in MTP is to expose a hierarchical file system present on the device. By making use of Associations in the communications protocol,

hierarchies may be represented without requiring any particular file name or path name conventions, or any understanding of the device file system.

Folder hierarchies are storage-specific, and are exposed in a bottom-up fashion using the **Parent Object** field in the **ObjectInfo** dataset. This field contains the **Object Handle** of the Association that "contains" the object. The **Parent Object** field is also identified by the appropriate object property (for more information, see the section on Object Properties).

Only objects of type Association may be referenced in the **Parent Object** field/property. By representing object hierarchies in a bottom-up way, associations are stateless with regard to which objects are their children, as only children identify their parents. In this way, object deletion does not result in having to re-establish the object hierarchy already communicated.

Objects that exist in the root of the storage shall contain a value of 0x00000000 in the **Parent Object** field/property.

A full path name may be reconstructed by traversing the parentage of a particular object, concatenating each **Filename** with an appropriate delimiter, in a file-system-specific way.

3.6.4 Associations and Object References

MTP enhances Associations using object references, which make generic folder Association objects more efficient. This feature can be used when an Association object represents a generic folder (Association Type = 0x0001), if the value of the **AssociationDesc** field is 0x00000001.

For such Associations, **GetObjectReferences** must return references to all of the objects contained in the folder. When objects are added to or deleted from such a folder (whether by the Initiator or Responder), the reference list shall be updated automatically.

3.7 MTP Extensibility

The vendor-extension mechanism described in the following section addresses two major scenarios. The first scenario is allowing a vendor to access the functionality of the USB-IF PTP extensions while still providing their own extensions, either pre-existing or created at a later date. The second scenario is enabling vendors that do not have an assigned VendorExtensionID to extend the protocol as they require.

This extension scheme allows multiple vendor-extension sets to be implemented concurrently, as long as their assigned datacodes do not overlap. It is the responsibility of the device vendor to avoid such conflicts.

3.7.1 Identifying Vendor Extension Support

Devices identify their support for vendor extensions using the **VendorExtensionDesc** field of the **DeviceInfo** dataset. The **VendorExtensionDesc** field is of datatype string, and provides a human-readable description of the supported extensions. Each supported extension is represented in the string value, with both the name of the extension set and the version number of the extension set. The name and version number of the extension set must follow a specific format:

- The name must be a valid internet domain name, owned and operated by the organization defining the extension set.
- The version number must have the traditional format of a Dewey decimal number.
- The name of the extension set is terminated with the colon character, and the version number is terminated with the semicolon character. A single space is required after each terminator character (':' and ';').

Example: "abc.com: 1.0; "

If several extension sets are implemented by the device, the series of names and version numbers of the extension sets are joined in the string value. The order of the extension sets within the value is not meaningful; the extension sets can be re-ordered within the value with no effect on the device implementation.

Example: "company1.com: 1.2; company2.com: 2.1.4; "

3.7.2 Linking to Extension Documentation and Specification

Extension set names and version numbers can be used to find human-readable documentation and machine-readable definitions on the Internet. The extension-set name and version number are placed into a template URL and used to reference both types of information. The following URL template is used to build the target URL:

"http://www.[name]/standards/protocols/mtp/[version]/

Where the extension set name replaces the [name] parameter, and the extension set version number replaces the [version] parameter. For example, the extension set for "abc.com: 1.1;" has an extension set name of "abc.com" and an extension set version number of "1.1". Replacing the [name] and [version] parameters in the URL template would produce the target URL:

"http://www.abc.com/standards/protocols/mtp/1.1/"

The type of the HTTP request determines which type of information is returned, assuming that the information is available on a public server. A request for HTML will

return the human-readable documentation describing the extension set, and a request for XML will return the machine-readable XML schema describing the extension set.

It is not required that all extensions be documented in this way, but it is very strongly recommended.

3.7.3 Allowed Datacode Ranges

Datacodes for vendor extensions to this extension set must fall within the ranges identified within each datacode definition as being reserved for vendor extensions to MTP. This is a subdivision of the standard vendor-extension ranges of PTP. If no vendor-extension range exists for the datacode that a vendor wants to extend, the vendor must attempt to determine an alternate implementation of the desired functionality or request a revision to this specification.

4 Communication Model

MTP follows a simple operation-data-response model for all communications. While the resulting functionality may be complex, it is entirely layered upon the core fundamentals described in this chapter.

4.1 Initiator/Responder Roles

MTP exchanges may only occur between two products at a time, and in each communication, one product acts as the Initiator and the other as the Responder. The Initiator is the product that initiates actions with the responder by sending operations to the Responder. The Responder may not initiate any actions, and may only send responses to operations sent by the Initiator or send events.

In this specification, the USB Host is the Initiator, and the USB Device is the Responder.

The Initiator must be able to enumerate and understand the contents of the Responder, handle and respond to events generated by the Responder, and control the flow of operations in the protocol.

Products expected to take on the Responder role include simple content-production devices, such as digital cameras or portable audio recorders, and simple content-display devices, such as personal information managers and portable audio players. Products expected to take on the Initiator role include personal computers, and products such as direct-print printers, designed primarily to connect with simple content-production devices.

4.2 Unidirectional Data Flow

The data flow in MTP is always unidirectional. When initiating an operation, data flows only from the Initiator to the Responder. When responding to the requested operation, the data flows only from the Responder to the Initiator. During the binary data-exchange phase, data may flow from the Responder to the Initiator or from the Initiator to the Responder, but never both. Bi-directional, binary data exchange must be performed by multiple operations.

4.3 MTP Transactions

Any Initiator-initiated action in MTP is performed in a transaction, a standard sequence of phases that provides the mechanism for action invocation with input parameters, responses with parameters, and binary data exchange. Note that the name "Transactions" is used for compatibility with the PTP specification, and must not be confused with "Transaction" as used in the USB 2.0 specification and related documents. If there is possibility of confusion, the term "MTP Transaction" will be used to refer to the

transactions defined in this document, and “USB Transaction” will be used to refer to the low-level transactions of the USB spec.

4.3.1 Transaction Synchronicity

All transactions in an MTP exchange are synchronous. That is, a new operation cannot be initiated until the previous operation has fully completed. If a device supports multiple sessions, it must manage each session to simulate synchronicity for each connected device.

Asynchronous operations must be simulated by separating the operation into an initiation, which begins the operation, and progress monitoring, through Responder-generated events sent while the operation is executed in the background on the device. If an operation is attempted while an asynchronous operation is processing which cannot be fulfilled due to the ongoing asynchronous operation, the responder shall respond with a Device_Busy failure code.

The only exception to this atomic process involves events. Events may be sent at any time, and are defined in such a manner as to allow them to be sent without interrupting an ongoing transaction. For more information about events, refer to section 4.8 Events.

4.3.2 Transaction Phases

A transaction in MTP consists of up to three phases: the Operation Request Phase, the Data Phase, and the Response Phase. The Operation Request Phase and Response Phase are identified together by sharing a **TransactionID**. The Data Phase is optional, and is placed between the two other phases of the transaction when used.

4.3.3 Transaction IDs

Transactions are identified using an unsigned 32-bit integer called a **TransactionID**. TransactionIDs are chosen by the Initiator, and should be chosen starting at 0x00000001 for the first operation initiated in a session and incremented by 1 for each successive transaction.

The **TransactionID** of 0xFFFFFFFF is invalid, and shall not be used by an Initiator. Instead, if a sequence of transactions results in the TransactionID incrementing to 0xFFFFFFFF, it shall instead wrap around to 0x00000001.

4.4 Sessions

MTP contains the concept of a session, which is a communications state in which a connection has persisted and a state has been maintained. Sessions serve two functional purposes in MTP: they provide a single context in which to reference objects, and they guarantee that the Responder device state has not changed without alerting the Initiator to that change.

An open session is not required for all operations, only for operations that require state context to execute or respond accurately. Any operation involving an Object Handles or StorageID is required to be session-aware. However, operations such as **GetDeviceInfo** do not involve any session-dependent information, and can be executed with or without an active session.

4.4.1 Opening and Closing Sessions

The Initiator opens a Session by sending the **OpenSession** operation.

A session may be ended by the Initiator by sending a **CloseSession** operation, is ended automatically when there is an interruption in communications between Initiator and Responder, or may be ended at any time by the Responder. If the Responder chooses to end a session, it shall indicate this by responding to the next session-aware operation with a Session_Not_Open response code.

4.4.2 Choosing Session IDs

A product may maintain multiple MTP sessions concurrently, and is not required to fill the same role in every session. It is thus required that the initiator identify the session in which it is acting when it sends any session-aware operation. Sessions are identified by a **SessionID**, a 32-bit unsigned integer, which is chosen by the Initiator and communicated in an **OpenSession** operation.

If an Initiator attempts to open a session on a device that already has an open session and that does not support multiple sessions, the Responder shall respond with a Session_Already_Open response, and pass the SessionID of the open session as the first parameter.

If an Initiator attempts to choose a SessionID that is already in use on a device that supports multiple concurrent sessions, the Responder shall respond with Session_Already_Open, and pass the chosen SessionID as the first parameter in the response.

If an Initiator attempts to open a session on a device that supports a finite number of multiple concurrent sessions, but that has already opened the maximum number of supported sessions, the Responder shall fail with a response code of Device_Busy.

The values 0x00000000 and 0xFFFFFFFF have special meanings for **SessionID**s. These meanings are described in relevant operations.

4.5 Operations

The Operation Request Phase initiates a transaction, and consists of the transmission of an **Operation Request** dataset from the Initiator to the Responder. The dataset identifies the operation that is being invoked by the initiator using an Operation Code, the context in which it is to be executed (session and transaction), and a limited set of parameters modifying the request.

4.5.1 Operation Datacodes

Operations are defined by a 16-bit datacode, which is included in the Operation Request Dataset as defined in section 4.5.2.

The most significant 5 bits have special meaning, as described in the following table.

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bits 10-0	Values	Datacode type
0	0	0	1	Any	Any	0x1000.. 0x1FFF	PTP Operation Code
1	0	0	1	0	Any	0x9000.. 0x97FF	MTP Vendor Extension Operation Code
1	0	0	1	1	Any	0x9800.. 0x9FFF	MTP Operation Code

All other values are reserved.

4.5.2 Operation Dataset

The layout of the Operation Dataset is provided for reference. Depending on the capabilities of the transport layer, some of these fields may be omitted at the interface between Initiator and Responder, and generated as needed for internal use. For example, the USB SIC transport omits the SessionID, because SIC only supports a single session. The USB SIC transport includes a length prefix, which allows unused parameters to be omitted at the USB layer.

Field	Size (bytes)	Datatype
Operation Code	2	UINT16
SessionID	4	UINT32
TransactionID	4	UINT32
Parameter 1	4	Any
Parameter 2	4	Any
Parameter 3	4	Any
Parameter 4	4	Any
Parameter 5	4	Any

4.5.3 Operation Parameters

Information required to act on an initiated operation may be passed as parameters in the Operation Request. This allows up to 5 parameters to be sent, each consisting of 32-bits or less.

Additional information may also be passed in a pre-defined dataset in the Data Phase of the transaction. Additional information about the Data Phase is available in section 4.6 Data Phases.

4.5.3.1 Operation Code

This identifies the operation being initiated by the device. For a list of these operations and their usages, refer to the appropriate appendix.

4.5.3.2 SessionID

This identifies the session in which this operation exists. For more information about sessions, refer to section 4.4 Sessions. If an operation occurs outside an active session, this field shall be set to 0x00000000.

4.5.3.3 TransactionID

This provides an identifier for the transaction initiated by this operation request. For more information, refer to section 4.3 MTP Transactions. This field shall contain 0x00000000 for operations that do not occur within a session.

4.5.3.4 Parameter *n*

These fields allow parameters to be passed along with an operation request. The meaning of the contents of these fields is context-specific, and depends on the Operation Code in the first field.

If a parameter is unused or marked as “None”, the parameter value is undefined and shall be silently ignored. If data exists in a parameter marked as “None”, a Responder shall return the response code Parameter_Not_Supported.

4.6 Data Phases

Following the Operation Request Phase is an optional Data Phase. The presence of this phase is determined by the Operation Code sent in the Operation Request Phase. If the Operation Request Phase results in an error state, the device shall still enter a Data Phase if the Operation Code in the Operation Request Phase indicates that one is required. If the Data Phase of an operation is I->R, and the responder enters an error state during or before transfer, the Responder shall complete the data phase as defined (possibly throwing away data) in order to indicate an appropriate error condition in the Response phase.

The Data Phase is used to pass any data that cannot be transferred using the parameters of the operation-request dataset. For some operation requests, this data will consist of datasets defined by this specification. For other operation requests, the data will be binary data exchanged for the purpose of storage on the receiving device. If the data being sent in the data phase is not a dataset defined by this specification, it must be considered opaque.

The actual transmission of data may involve wrapping the data to be sent in a container format, or breaking it apart into packets to be reassembled upon receipt.

Note that when SIC class is being used for a data transport, a container is used.

4.7 Responses

Following every operation, the responder returns a response dataset as defined in section 4.7.2 Response Dataset. This response dataset contains up to five 32-bit parameters and a **ResponseCode** indicating the result of the operation.

ResponseCodes other than OK indicate that the operation did not complete.

4.7.1 Response Datacodes

Responses are identified by their ResponseCode, which is included in the response dataset as defined in section 4.7.2 Response Dataset. All ResponseCodes are 16-bit integers, with the most significant five bits having special meaning, as described in the following table.

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bits 10-0	Values	Datacode type
0	0	1	0	Any	Any	0x2000.. 0x2FFF	PTP ResponseCodes
1	0	1	0	0	Any	0xA000.. 0xA7FF	Vendor Extension ResponseCodes
1	0	1	0	1	Any	0xA800.. 0xAFFF	MTP ResponseCodes

4.7.2 Response Dataset

The layout of the Response Dataset is provided for reference. Depending on the capabilities of the transport layer, some of these fields may be omitted at the interface between Initiator and Responder, and generated as needed for internal use. For example, the USB SIC transport omits the SessionID, because SIC only supports a single session. The USB SIC transport includes a length prefix, which allows unused parameters to be omitted at the USB layer.

Field	Size (bytes)	Format
ResponseCode	2	UINT16
SessionID	4	UINT32
TransactionID	4	UINT32
Parameter 1	4	Any
Parameter 2	4	Any
Parameter 3	4	Any
Parameter 4	4	Any
Parameter 5	4	Any

4.7.3 Response Parameters

Responses may include up to five 32-bit parameters, and all devices must include and accommodate all parameters. When a parameter is not used for a given response, the corresponding field shall be set to 0x00000000.

4.7.3.1 ResponseCode

This datacode identifies the result of the requested operation. Further definition of allowed responses can be found in Appendix F – Responses.

4.7.3.2 SessionID

This field identifies the session in which this operation exists. For more information, refer to section 4.4 Sessions. If an operation occurs outside of an active session, this field shall

be set to 0x00000000. The value of this field shall be identical to the value in the Operation Request dataset that was received by the Responder in this transaction.

4.7.3.3 TransactionID

This field provides an identifier for the transaction initiated by this operation request. For more information, refer to section 4.3 MTP Transactions. This field shall contain 0x00000000 for operations that do not occur within a session.

4.7.3.4 Parameter *n*

These fields allow parameters to be passed along with a response code. The meaning of the contents of these fields is context-specific, and depends on the Response Code in the first field.

If a parameter is unused or marked as "None", the parameter value is undefined and should be silently ignored.

4.8 Events

This section describes events, their datasets and their usages. Event support is a mandatory part of MTP.

Although events may be sent by either the Initiator or the Responder, they are primarily sent by the Responder as a way of proactively transmitting information or alerts. Unlike operations, events are not acknowledged, and need not be acted upon.

Events are not intended to convey information beyond the notification of an event. If an event is indicating the change of state on the device, the event dataset will contain only enough information to enable the device receiving the event to determine the nature of the state change with minimal effort through normal methods.

Events may be implemented in multiple steps, by first indicating the presence of an event, and then requiring a round-trip to retrieve it. When that is the case, the event code shall be sent in the initial event-indication dataset, and the remaining data shall be retrievable upon request.

Events are only sent to Initiators or Responders that are connected with an open session. For more information on sessions, refer to section 4.4 Sessions.

4.8.1 Event Datacodes

All events are identified by their Event Code, which is a 16-bit datacode. The most significant 5 bits have special meaning, as defined in the following table.

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bits 10-0	Datacode type
0	1	0	0	Any	Any	PTP Event Codes
1	1	0	0	0	Any	Vendor extension Event Code
1	1	0	0	1	Any	MTP Event Codes

4.8.2 Event Dataset

Events are communicated in the form of an event dataset, which consists of the minimum required information to act on the event. In general, the data contained in this dataset is minimized, and if the event code alone is insufficient, the device receiving the event must probe the sending device for more information after receiving the event.

Field	Size (bytes)	Format
Event Code	2	UINT16
SessionID	4	UINT32
TransactionID	4	UINT32
Parameter 1	4	Any
Parameter 2	4	Any
Parameter 3	4	Any

4.8.3 Event Parameters

4.8.3.1 Event Code

This field identifies the event being indicated by this dataset. A listing of event codes and their meanings can be found in the appropriate Appendix of this specification.

4.8.3.2 SessionID

This field identifies the SessionID for which the event is relevant. If this event is relevant to all current sessions, it shall contain 0xFFFFFFFF. If a device receives an event for which the SessionID does not match a session that the receiving device currently has open, and that does not contain 0xFFFFFFFF, then the event shall be ignored.

For more information on sessions, refer to section 4.4 Sessions.

4.8.3.3 TransactionID

If the event corresponds to a previously or currently occurring transaction, this field shall contain that transaction's TransactionID. For more information, refer to section 4.3 MTP Transactions.

4.8.3.4 Parameter n

This field holds a parameter that can be passed with this event. Events may have at most three parameters, and all must occupy 32 bits. The interpretation of event parameters depends upon the event with which they are being sent.

If a parameter is unused or marked as "None", the parameter value is undefined and should be silently ignored.

4.8.4 Asynchronous Event Support

As explained in section 2.3 Asynchronous Events, events are required to be communicated asynchronously with data transmission or operation transactions.

4.8.4.1 Interleaving Events

If a dedicated event channel exists, the protocol for that channel must define a process by which events may be interleaved within a data stream during a transaction. It may be assumed that the Operation Request Phase and Response Phase (For more information, see sections 4.5 Operations and 4.7 Responses.) are atomic, but the Data Phase must allow for events to be communicated in either direction without interrupting data transfer. For more information about the data phase, refer to section 4.3.2 Transaction Phases and section 4.6 Data Phases.

Care should be taken to ensure sufficient responsiveness (in general, responsiveness should be prioritized above data-transfer speed).

5 Device Model

MTP is a protocol designed to represent an abstracted view of a device which can be loosely defined by the following criteria:

- It has storage.
- It interacts with its own storage.
- It fulfils its primary purpose while not in an MTP session.
- It frequently connects to other devices using MTP in order to exchange and update content.

5.1 Device Representation

In MTP, a device has equal prominence within the protocol as its contents. Understanding the capabilities and properties of a device enables a number of important scenarios above and beyond simple data transfer.
Examples of enabled scenarios include:

- Rich UI representation of a connected device
- Matching content to device capabilities
- Meta-functionality on objects, such as DRM
- Device state awareness, such as battery level or capture settings
- Device command and control
- Etc.

These scenarios are implemented by a combination of a standard device-describing dataset (the DeviceInfo dataset) to provide basic device capabilities, which is always present and implicit in MTP functionality; and flexible and extensible device properties. Both are discussed in more detail below.

5.1.1 DeviceInfo Dataset

The DeviceInfo dataset is used to provide a description of the device. This dataset can be obtained using the **GetDeviceInfo** operation without first initiating a session, and is mostly static. If any value in this dataset is changed while a session is active, a **DeviceInfoChanged** event shall be issued to each connected Initiator, and each Initiator must re-acquire the DeviceInfo dataset to determine the updated values.

An example of a situation where the DeviceInfo dataset could change within a session is as a reaction to a change in the Functional Mode of the device. A device may enter a "sleep" state where it has a limited (but sufficient) set of enabled MTP operations and functionality. When such a state is entered, a **DeviceInfoChanged** event is issued to each active session to alert them to the changed functionality of the device.

Dataset field	Field order	Size (bytes)	Datatype
Standard Version	1	2	UINT16
MTP Vendor Extension ID	2	4	UINT32
MTP Version	3	2	UINT16
MTP Extensions	4	Variable	String
Functional Mode	5	2	UINT16
Operations Supported	6	Variable	Operation Code Array
Events Supported	7	Variable	Event Code Array
Device Properties Supported	8	Variable	Device Property Code Array
Capture Formats	9	Variable	Object Format Code Array
Playback Formats	10	Variable	Object Format Code Array
Manufacturer	11	Variable	String
Model	12	Variable	String
Device Version	13	Variable	String
Serial Number	14	Variable	String

5.1.1.1 Standard Version

This identifies the PTP version this device can support in hundredths. For MTP devices implemented under this specification, this shall contain the value 100 (representing 1.00).

5.1.1.2 MTP Vendor ExtensionID

This identifies the PTP vendor-extension version in use by this device. For MTP devices implemented under this specification, this shall contain the value 0xFFFFFFFF.

5.1.1.3 MTP Version

This identifies the version of the MTP standard this device supports. It is expressed in hundredths. The final version of this specification will identify the correct value to place in this field.

5.1.1.4 MTP Extensions

This string is used to identify any extension sets applied to MTP, and is discussed in length later in this specification.

5.1.1.5 Functional Mode

Modes allow the device to express different states with different capabilities. If the device supports only one mode, this field shall contain the value 0x00000000.

Value	Description
0x0000	Standard mode
0x0001	Sleep state
All other values with bit 15 set to 0	Reserved
0xC001	Non-responsive playback
0xC002	Responsive playback
All other values with bit 15 set to 1 and bit 14 set to 0	MTP vendor extension
All other values with bit 15 set to 1 and bit 14 set to 1	MTP-defined

The current functional mode is also contained in a device property. In order to change the functional mode of the device, a session must be opened and the appropriate device property updated (if allowed). More information about device properties is available later in this document.

5.1.1.6 Operations Supported

This field identifies by datacode all operations that this device supports in the current functional mode.

5.1.1.7 Events Supported

This field identifies by datacode all events that this device can generate in the current functional mode.

5.1.1.8 Device Properties Supported

This field identifies by datacode all device properties that this device supports in the current functional mode.

5.1.1.9 Capture Formats

This field identifies by datacode the object format codes for each format that this device can generate independently (that is, without the content being placed on the device).

5.1.1.10 Playback Formats

This field identifies by datacode the object format codes for each format that this device can understand and parse if placed on the device.

If the device can carry unidentified binary objects without understanding them, it shall indicate this by including the Undefined Object (0x3000) code in its Playback Formats.

5.1.1.11 Manufacturer

This optional string is a human-readable string identifying the manufacturer of this device.

5.1.1.12 Model
This optional string is a human-readable string identifying the model of this device.

5.1.1.13 Device Version
This optional string identifies the software or firmware version of this device in a vendor-specific format.

5.1.1.14 Serial Number
This string is required, and contains the MTP function's serial number. Serial numbers are required to be unique among all MTP functions sharing identical Model and Device Version fields (this field was optional in the PTP specification, but is required in MTP). The serial number should be the device's unique serial number such as the one typically printed on the device.

The serial number shall be a 32 character hexadecimal string for legacy compatibility reasons. This string must be exactly 32 characters, including any leading 0s, and does not require any prefix to identify it as hexadecimal (such as '0x').

5.1.2 Device Properties
This section describes device properties. Device-property support is a mandatory part of PTP, and remains unchanged in MTP beyond additional, added device properties.

Device properties identify settings or state conditions of the device, and are not linked to any data objects on the device. Objects on the device are described using Object Properties, which are discussed further in section 5.3.2 Object Properties.

Device Properties may be read-only or read-write, and serve different functions depending on the context in which they are used. A single device may have only one set of device properties, and they must be the same across all sessions and connections.

5.1.2.1 Device Property Describing Dataset
Device properties are defined by their **DevicePropDesc** dataset, which can be retrieved with the **GetDevicePropDesc** operation.

The **DevicePropDesc** dataset includes the device property value, read/write settings for the property, a default value and, where relevant, any restrictions on allowed values.

Restrictions on the allowed values of a device property are communicated using additional fields following the core dataset. The format of the additional forms is determined by a flag in the sixth field, which enumerates allowed forms.

Field name	Field order	Size (bytes)	Datatype	Description
Device Property Code	1	2	UINT16	A specific device property code.
Datatype	2	2	UINT16	Identifies the data type code of the property, as defined in section 3.2 Simple Types.
Get/Set	3	1	UINT8	Indicates whether the property is read-only (Get), or read-write (Get/Set). 0x00 Get 0x01 Get/Set
Factory Default Value	4	DTS	DTS	Identifies the value of the factory default for the property.
Current Value	5	DTS	DTS	Identifies the current value of this property.
Form Flag	6	1	UINT8	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 depends on the Form Flag, and is absent if Form Flag = 0x00.

5.1.2.1.1 Range Form

Range form				
Field name	**Field order**	**Size (bytes)**	**Datatype**	**Description**
Minimum Value	7	DTS	DTS	Minimum value supported by this property.
Maximum Value	8	DTS	DTS	Maximum value supported by this property.
Step Size	9	DTS	DTS	A particular vendor's device shall support all values of a property defined by Minimum

				Value + N x Step Size, which is less than or equal to Maximum Value where N= 0 to a vendor-defined maximum.

5.1.2.1.2 Enumeration Form

Enumeration form				
Field name	**Field order**	**Size (bytes)**	**Datatype**	**Description**
Number Of Values	7	2	UINT16	This field indicates the number of values of size DTS of the particular property supported by the device.
Supported Value 1	8	DTS	DTS	A particular vendor's device shall support this value of the property.
Supported Value 2	9	DTS	DTS	A particular vendor's device shall support this value of the property.
Supported Value 3	10	DTS	DTS	A particular vendor's device shall support this value of the property.
…	…	…	…	…
Supported Value M	M+7	DTS	DTS	A particular vendor's device shall support this value of the property.

5.1.2.2 Retrieving Device Properties

Device Properties may be retrieved by one of two methods: They may be retrieved as a part of the Device Property Description Dataset returned by the **GetDevicePropDesc** operation, or they may be retrieved in a more-streamlined fashion by the **GetDevicePropValue** operation.

If both operations are supported by a Responder, the Initiator may use the **GetDevicePropValue** whenever the additional information contained in the **DevicePropDes**c dataset is not required. If a Responder is to optimize device-property retrieval, it shall enable and implement the **GetDevicePropValue** operation. Similarly, if an Initiator wishes to be performance-conscious when retrieving device properties, it should use the **GetDevicePropValue** operation if implemented. The

GetDevicePropDesc/GetDevicePropValue operation should only be called when information other than the **DevicePropDesc** dataset is required.

5.1.2.3 Setting Device Properties

Device property values may be set by the **SetDevicePropValue** operation, or may be updated as a result of changes to the state of the device. It is also permitted for a device property to be static, to provide information about the device that is not contained by the **DeviceInfo** dataset.

When a device property value is changed by a **SetDevicePropValue** operation request, all connected Initiators shall be alerted to the change with a **DevicePropChanged** event, except the Initaitor that directly caused the change. If a device property value is updated by any mechanism except the **SetDevicePropValue** operation, then all connected Initiators shall receive the **DevicePropChanged** event, including any Initiators that may have indirectly caused the property to be changed.

Device properties shall all be set atomically, and the act of setting one device property does not imply a change to any other device property. In cases where device property values are inherently intertwined, this specification combines those values into a single property where possible. For example, Width and Height are combined into the **Image Size** property.

If updating one property changes the indicated allowed values for another property, such as if increasing the **Image Size** reduced the allowed **Bit Depth** settings, this should be indicated using the **DevicePropChanged** event.

5.1.2.4 Device Properties as Device Control

Device Properties may be read-only or read-write. In the case where device properties identify the current functional state of the device, the state may be changed through the use of a writeable device property.

An example of a Device Property used for functional control of the device is the **Digital Zoom** device property, which identifies not only the current Digital Zoom setting, but also allows the initiator to set a new Digital Zoom setting. Another example is the **Functional Mode** device property.

5.2 Storage Representation

MTP devices generally include a substantial amount of persistent data storage, either contained in the device or on a removable storage medium. This section provides additional information about the required representation of that storage.

5.2.1 Storage IDs

Storages are identified in MTP using a 32-bit unsigned integer (UINT32), called a **StorageID**. The **StorageID** is subdivided into two halves, the most-significant 16 bits and the least-significant 16 bits. The most-significant 16 bits identify a physical storage location, such as a removable memory card or an internal memory bank. The least-significant 16 bits identify a logical partition of that physical storage.

Devices may contain zero or more physical storages, and each physical storage may have zero or more logical storages. Each physical storage is defined by a unique 16-bit code occupying the most-significant 16 bits of the **StorageID**. If two **StorageIDs** contain an identical, top-most 16 bits, they are assumed to exist on the same physical component of the device. The upper 16-bits of the **StorageID** may contain any values except 0x0000 and 0xFFFF, which may have special meaning, depending on the lower 16 bits.

If a physical storage contains no logical storages, it shall be represented using a single **StorageID** in which the least-significant 16-bit segment contains the value 0x0000. Any storage for which the lower 16 bits are 0x0000 is assumed to neither contain any data nor be able to have data written to it.

If a physical storage contains one or more logical storages, each storage must contain the same top-most 16-bit segment, to indicate that it is located in the same physical location.

Logical stores are defined by the lower 16-bits of the **StorageID**. Each logical storage on a physical storage must be identified by a unique 16-bit segment. The lower 16 bits of the **StorageID** may contain any value but 0x0000 (which has special meaning as outlined previously) and 0xFFFF (which may have special meaning depending on the upper-most 16 bits).

A **StorageID** of 0x00000000 or 0xFFFFFFFF has special meaning, depending on context, such as "All Storages" or "default store", and does not refer to an actual storage. The meaning is explained with the operation/response/event where it is used.

StorageIDs shall not be assumed to persist between sessions.

5.2.2 StorageInfo Dataset Description

This dataset describes a storage contained in a device.

Dataset field	Field order	Length (bytes)	Datatype
Storage Type	1	2	UINT16
Filesystem Type	2	2	UINT16
Access Capability	3	2	UINT16
Max Capacity	4	8	UINT64
Free Space In Bytes	5	8	UINT64
***Free Space In Objects**	6	4	UINT32
Storage Description	7	Variable	String
Volume Identifier	8	Variable	String

5.2.2.1 Storage Type

This field identifies the physical nature of the storage described by this dataset. If the Storage Type is read-only memory (0x0001 or 0x0002), it supersedes any protection status indicated by the Access Capability field.

Allowed values are shown in the following table.

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

5.2.2.2 Filesystem Type

This field identifies the logical file system in use on this storage. This field may be used to define the file-naming conventions or directory structure conventions in use on this storage, as well as indicate support for a hierarchical file system.

Allowed values are shown in the following table.

Code Value	Description
0x0000	Undefined
0x0001	Generic flat
0x0002	Generic hierarchical
0x0003	DCF
All other values with bit 15 set to 0	Reserved
All other values with bit 15 set to 1 and bit 14 set to 0	MTP vendor extension
All other values with bit 15 set to 1 and bit 14 set to 1	MTP-defined

5.2.2.3 Access Capability

This field identifies any write-protection that globally affects this storage (this supersedes any protection status on individual objects). If the Storage Type field indicates that this storage is defined as ROM (0x0001 or 0x0002), this field must contain the value 0x0001 (read-only without object deletion).

Allowed values are shown in the following table.

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

5.2.2.4 Max Capacity

This field identifies the maximum capacity in bytes of this storage. If this storage can be written to, that is, the storage type is not read-only and the access capability is read-write, then this field must contain an accurate value. If the storage is read-only, this field is optional.

5.2.2.5 Free Space In Bytes

This field indicates how much space remains to be written to on the drive. If this storage can be written to, that is, the storage type is not read-only and the access capability is read-write, then this field must contain an accurate value. If the storage type is read-only, this field is optional. If Free Space In Bytes does not apply to this device or this storage, and it can be written to, this field may contain a value of 0xFFFFFFFF, and the Free Space In Objects field may be used instead.

5.2.2.6 Free Space In Objects

This field indicates how many additional objects may be written to this device. This field shall only be used if there is a reasonable expectation that the number of objects that remain to be written can be accurately predicted (for instance, if the device contains only one object type of fixed size.).
If this field is not used, it shall be set to 0xFFFFFFFF.

5.2.2.7 Storage Description

This optional field contains a human-readable string identifying this storage, such as "256Mb SD Card" or "20Gb HDD". If unused, it shall contain an empty string.

5.2.2.8 Volume Identifier

This field contains a unique, programmatically relevant volume identifier, such as a serial number. This field may be up to 255 characters long, however, only the first 128 characters will be used to identify the device, and these first 128 characters must be unique for all storages. If this field does not contain a string in which the first 128 characters are unique, it must contain an empty string.

5.2.3 Defining Access Restrictions

If the device has restrictions on allowed operations on the content of the device, such as read-only, or read-deletion only, this shall be indicated in the AccessCapability field of the **StorageInfo** dataset. For more information, refer to section 5.2.2 StorageInfo Dataset Description.

5.3 Content Representation

In MTP, all contents of a device are represented as objects. Objects are abstract containers for a variety of data, which each represent an atomic element of information. Examples of objects include:

- Image Files
- Audio/Video Files
- Contacts
- Calendar/Task Items
- Generic Binary Files
- Etc.

Objects are comprised of four parts: the object's binary data, the ObjectInfo dataset, Object Properties and Object References. Each component serves a different purpose, and together they facilitate broad interoperability and a rich enumeration experience.

The ObjectInfo Dataset is a standard fixed dataset available for every object, and provides basic information about an object. This information includes the object type, object size, etc, and represents the information required for every object in order to enable basic object management. The ObjectInfo dataset was originally defined in PTP, and has been largely replaced in MTP by Object Properties.

Object Properties provide a flexible and extensible way of representing object metadata. They may be used to describe a device in either (or both) a machine-readable or human-readable way, and serve to not only describe the actual content of the object but also to indirectly indicate the various structures a particular object format can take.

Finally, Object References provide an internal referencing feature within MTP, allowing objects to associate themselves with other objects, a feature which would otherwise be impossible without a standard persistent addressing mechanism.

5.3.1 ObjectInfo Dataset Description

The **ObjectInfo** dataset provides an overview of the core properties of an object. These properties are also retrievable as **Object Properties** (detailed later in this document), and must be accessible through both mechanisms.

The contents of the **ObjectInfo** dataset were originally defined for imaging-centric devices, and continue to be used for that purpose. For devices to be compatible with existing PTP implementations, it is very important that this dataset be accurately filled out in accordance with the PTP specification. For more information, please refer to that document.

Many fields in the **ObjectInfo** dataset do not apply to non-imaging objects; these fields shall be set to zero when not in use.

MTP still requires the use of the **ObjectInfo** dataset in various operations that require an encapsulated view of core file attributes, such as when sending a new object to the device. The fields that are required for the MTP **SendObjectInfo** operation are specifically called out in the following table.

Fields marked with a * are supplanted in MTP with new mechanisms, but must be implemented if PTP-compatibility is desired. Refer to the PTP specification for implementation details for these fields.

Dataset field	Field order	Size (bytes)	Datatype	Required for SendObjectInfo
StorageID	1	4	StorageID	No
Object Format	2	2	ObjectFormatCode	Yes
Protection Status	3	2	UINT16	No
Object Compressed Size	4	4	UINT32	Yes
***Thumb Format**	5	2	ObjectFormatCode	No
***Thumb Compressed Size**	6	4	UINT32	No
***Thumb Pix Width**	7	4	UINT32	No
***Thumb Pix Height**	8	4	UINT32	No
Image Pix Width	9	4	UINT32	No
Image Pix Height	10	4	UINT32	No
Image Bit Depth	11	4	UINT32	No
Parent Object	12	4	ObjectHandle	No
Association Type	13	2	Association Code	Yes
Association Description	14	4	AssociationDesc	Yes
***Sequence Number**	15	4	UINT32	No

Filename	16	Variable	String	Yes
Date Created	17	Variable	DateTime String	No
Date Modified	18	Variable	DateTime String	No
Keywords	19	Variable	String	No

5.3.1.1 StorageID

The device storage on which the object defined by this dataset is located. For more information about StorageIDs, refer to section 5.2.1 "Storage IDs".

5.3.1.2 ObjectFormat

Every object format type is identified by an ObjectFormatCode. Refer to section 3.5 Object Formats for more information about Object Formats.

5.3.1.3 Protection Status

An optional field that identifies the write-protection status of a data object. This field may be updated using the SetProtection operation.

If the device does not support object protection, this field should always contain 0x0000, the SetProtection operation should not be supported, and the only allowed value identified in the ObjectPropDesc field should be 0x0000.

Read-only protection may modified with the SetObjectProtection operation if allowed by the device. Read-only protection may not be modified by mechanisms used to set Object Properties.

Allowed values are shown in the following table:

Value	Description
0x0000	No protection
0x0001	Read-only
0x8002	Read-only data
0x8003	Non-transferable data
All other values	Reserved

No Protection: This object has no protection; it may be modified or deleted arbitrarily and its properties may be modified freely.

Read-only: This object cannot be deleted or modified; none of the properties of this object can be modified by the initiator. (However, properties can be modified by the device that contains the object.)

Read-only data: This object's binary component cannot be deleted or modified; however; any object properties may be modified if allowed by the object property constraints.

Non-transferable data: This object's properties may be read and modified, and it may be moved or deleted on the device, but this object's binary data may not be retrieved from the device using a **GetObject** operation.

This property identifies the write-protection status of the binary component of the data object.

The allowed values for this property for a device should be enumerated in ObjectPropDesc dataset and are defined as follows:

Value	Description
0x0000	No Protection This object has no protection, it may be modified or deleted arbitrarily, and its properties may be modified freely.
0x0001	Read-only This object cannot be deleted or modified, and none of the properties of this object can be modified by the initiator. (Properties may be modified by the device which contains the object however.)
0x8002	Read-only data This object's binary component cannot be deleted or modified, however any object properties may be modified if allowed by the object property constraints.
0x8003	Non-transferrable data This object's properties may be read and modified, and it may be moved or deleted on the device, but this object's binary data may not be retrived from the device using a GetObject operation.
0x0002-0x7FFF	Reserved for PTP
0x8004-0x8BFF	Reserved for MTP
0x8C00-0xFFFF	MTP Vendor Extension Range

This property may be indirectly set using the SetObjectProtection operation.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC03
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	0x00 (GET)
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device Defined
FormFlag	6	1	UINT8	0x02 Enumeration Form

5.3.1.4 Object Compressed Size

The size of the data component of the object in bytes. If the object is larger than 2^32 bytes in size (4GB), this field shall contain a value of 0xFFFFFFFF.

5.3.1.5 *Thumb Format, *Thumb Compressed Size, *Thumb Pix Width, *Thumb Pix Height

These fields provide information about thumbnails of images on the device. In MTP, thumbnail functionality has been supplanted with the Representative Sample Object Properties, and in most cases these values shall be duplicated there.

If PTP-compatibility is desired, these fields shall be supported for all image objects, and the associated operations shall be implemented.

5.3.1.6 Image Pix Width

If the defined object is an image, this field identifies the width in pixels of that image. This information is also available in MTP through the object property mechanism, but shall be implemented here to maintain PTP-compatibility if desired.

5.3.1.7 Image Pix Height

If the defined object is an image, this field identifies the height in pixels of that image. This information is also available in MTP through the object property mechanism, but shall be implemented here to maintain PTP-compatibility if desired.

5.3.1.8 Image Bit Depth

If the defined object is an image, this field identifies the bit depth of that image. This information is also available in MTP through the object property mechanism, but shall be implemented here to maintain PTP-compatibility if desired.

5.3.1.9 Parent Object

If this object exists in a hierarchy, this field contains the Object Handle of the parent of this object. Only objects of type Association may be identified in this field. If this object is in the root or the device does not support Associations, this field shall be set to 0x00000000.

More information about Associations is contained in section 3.6 Associations.

5.3.1.10 Association Type

This field is only used for objects of type Association, and indicates what type of Association it is. In MTP, the Generic Folder Association type is most commonly used, which has an Association Type code of 0x0001.

More information about Associations is contained in section 3.6 Associations.

5.3.1.11 AssociationDesc

This field is used only for objects of type Association, and provides additional information about the implementation of this particular type.

More information about Associations is contained in section 3.6 Associations.

5.3.1.12 Sequence Number

This field is used in PTP to further define certain non-folder types of associations. It is not generally used in MTP, but should be implemented by any Responder that desires the provided functionality and that wishes to interact with PTP Initiators.

More information about the use of Sequence Numbers in Associations is contained in section 3.6 Associations.

5.3.1.13 Filename

A string containing the file name of this object, without any directory or file system information. This string is also accessible and defined via an Object Property, and restrictions on its format may be identified in the Object Property Description for this object property.

5.3.1.14 Date Created

This field identifies the creation date of this object, and uses the DateTime string as described in section 3.2.5 DateTime.

5.3.1.15 Date Modified

This field identifies the date when this object was last modified, and uses the DateTime string described in section 3.2.5 DateTime.

5.3.1.16 Keywords

This field contains keywords associated with the object. Keywords shall be delimited by a single space: " ". If a given keyword contains a space, the space shall be replaced with an underscore character: "_".

5.3.2 Object Properties

Object properties provide a mechanism for exchanging object-describing metadata separate from the objects themselves. The primary benefit of this functionality is to permit the rapid enumeration of large storages, regardless of the file-system.

In PTP, there is an existing mechanism for describing an object, the **ObjectInfo** dataset (PTP Specification, section 5.5.2). This is a static and non-extensible dataset containing basic information about the object, and assumes that the object being described is an image object.

There is also an extensible property system in device properties (PTP Specification, section 13). These properties are notable, as they define the format their values can take, which allows for much safer device management. By restricting property values and formats at the protocol level, the responder has a much simpler time parsing and understanding that metadata for its own use.

Object properties combine these two concepts. They provide information about objects on the device, and specify the values they can contain.

5.3.2.1 Requirements for Object Property Support

A device that will implement object properties must support the following operations:

- **GetObjectPropsSupported**
- **GetObjectPropValue**
- **GetObjectPropDesc**

A corollary of this is that all devices that support these minimum requirements are assumed to support object properties as a primary method of retrieving object metadata.

5.3.2.2 Identifying Object Property Support

Devices that identify their support for the operations listed previously in their **DeviceInfo** dataset are considered to support object properties, and will be treated as such by initiators that understand the USB-IF PTP extension set. Devices that do not support the minimum requirements listed previously will not be considered to support object properties.

Object properties are format-specific. That is, for all objects on a device of the same format, the same set of object properties will be supported. Discovering which object properties are applied to a given object format is accomplished by the **GetObjectPropsSupported** operation, with the format code passed as the argument in the table in the next section.

The **GetObjectPropsSupported** operation returns an **ObjectPropCode** array of supported object properties for the object format indicated in the first parameter. More details about the **GetObjectPropsSupported** operation are available in the appropriate appendix of this specification.

5.3.2.3 Defining Object Properties

Object properties are defined by an **ObjectPropDesc** dataset much in the same way that device properties are defined by their **DevicePropDesc** dataset (PTP Specification, section 13.3.3).

Before an object property is queried for the first time, its **ObjectPropDesc** should be retrieved from the device by using the **GetObjectPropDesc** operation. The **ObjectPropDesc** dataset is defined in Table 5-1.

The **GetObjectPropDesc** operation returns the appropriate Property Describing Dataset, indicated in the first parameter as defined for the Object Format indicated in the second parameter. More details about the **GetObjectPropDesc** operation are available in Appendix D.2.30 GetObjectPropDesc.

A base set of object properties are defined in the MTP specification, but many aspects of an object property need to be defined by the device for its particular implementation, particularly properties that the device will use for its own operations.

The **ObjectPropDesc** dataset is formatted as shown in the following table:

Table 5-1. ObjectPropDesc dataset

Field name	Field order	Size (bytes)	Datatype	Description
Property Code	1	2	UINT16	A specific ObjectPropCode

				identifying this property.
Datatype	2	2	UINT16	This field identifies the datatype code of the property.
Get/Set	3	1	UINT8	This field indicates whether the property is read-only (Get), or read-write (Get/Set). 0x00 Get 0x01 Get/Set
Default Value	4	DTS	DTS	This field identifies the value of the factory default for the property.
Group Code	5	4	UINT32	This field identifies the retrieval group this property belongs to.
Form Flag	6	1	UINT8	This field indicates the format of the next field. 0x00 None 0x01 Range form 0x02 Enumeration form 0x03 DateTime form 0x04 Fixed-length Array form 0x05 Regular Expression form 0x06 ByteArray form 0xFF LongString form
FORM	N/A	<variable>	-	This dataset depends on the Form Flag, and is absent if Form Flag = 0x00.

5.3.2.3.1 Range Form

Field name	Field order	Size (bytes)	Datatype	Description
MinimumValue	7	DTS	DTS	Minimum value supported by this property.
MaximumValue	8	DTS	DTS	Maximum value supported by this property.
Step Size	9	DTS	DTS	The property shall support all values defined by MinimumValue+N*StepSize, which is less than or equal to MaximumValue.

5.3.2.3.2 Enumeration Form

Field name	Field order	Size (bytes)	Datatype	Description
NumberOfValues	7	2	UINT16	The number of values, of size DTS, supported by the property. These shall be listed in order of preference if applicable.
SupportedValue1	8	DTS	-	The property shall support this value.
SupportedValue2	9	DTS	-	The property shall support this value.
SupportedValueM	7+M	DTS	-	The property shall support this value.

5.3.2.3.3 DateTime Form

Properties that have the DateTime form have no additional fields. 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 to 12), *DD* is the day of the month (01 to 31), *T* is a constant character, *hh* is the hours since midnight (00 to 23), *mm* is the minutes past the hour (00 to 59), 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.

5.3.2.3.4 Fixed-length Array Form

Field name	Field order	Size (bytes)	Datatype	Description
Length	7	2	UINT16	A 16-bit unsigned integer giving the number of elements which must be included in the array contained by this property. All properties which have this form must contain an array.

5.3.2.3.5 Regular Expression Form

Field name	Field order	Size (bytes)	Datatype	Description
RegEx	7	-	String	A regular expression that must exactly generate the value of this property. A standardized syntax for regular expressions as they are used in MTP is available at: http://msdn2.microsoft.com/en-us/library/1400241x.aspx (page title: "Regular Expression Syntax (Scripting)")

5.3.2.3.6 ByteArray Form

Field name	Field order
MaxLength	The maximum length of the ByteArray that may be contained by this property. The device shall accept any property values that have a NumElements value equal to or less than this field. Properties that have a ByteArray form must contain an AUINT8 datatype, which contains an array of bytes.

5.3.2.3.7 LongString Form

Field name	Field order
MaxLength	The maximum length of the LongString that may be contained by this property. The device shall accept any property values that have a NumElements value equal to or less than this field. Properties that have a LongString form must contain an AUINT16 datatype, which contains characters encoded in 2-byte Unicode characters, as described in ISO10646.

5.3.2.4 Retrieving Object Properties

When information about an object is required, the properties can be retrieved one at a time using the **GetObjectPropValue** operation, which returns the current value of an object property. A complete description of the **GetObjectPropValue** operation can be found in the appropriate appendix of this specification.

5.3.2.5 Setting Object Properties

If a device supports the setting of object properties, which it indicates by supporting the **SetObjectPropValue** operation, then properties can be identified as settable by setting the **Get/Set** field in the **ObjectPropDesc** dataset. Those properties can be updated to any value that satisfies the constraints defined in its **ObjectPropDesc** dataset.

The **SetObjectPropValue** operation sets the current value of the object property indicated by parameter 2 for the object indicated by parameter 1 to the value indicated in the data phase of the operation. The format of the property value object sent in the data phase can be determined by the DatatypeCode field of the property's ObjectPropDesc dataset. If the property is not settable, the response Access_Denied shall be returned. If the value is not allowed by the device, Invalid_ObjectProp_Value shall be returned. If the format or size of the property value is incorrect, Invalid_ObjectProp_Format shall be returned.

The **SetObjectPropValue** operation is fully defined in the appropriate appendix of this specification.

5.3.2.6 Required Object Properties

Devices which implement Object Properties must implement as Object Properties the properties which map to the required fields of the ObjectInfo dataset, as well as certain properties which are required for effective functioning of the protocol. Required properties for all object formats include:

- StorageID
- ObjectFormat
- ObjectCompressedSize

- Persistent Unique Object Identifier
- Name

Devices which support file systems through the use of associations must support the following object properties:

- ParentObject

5.3.2.7 Optimizing Object Properties

The development of object properties allows considerable improvements in functionality over the static descriptive set of metadata exposed in the ObjectInfo dataset, however, it comes at the cost of a dramatically increased number of queries to the device. This has been mitigated in the Basic MTP specification by defining object-property retrieval groups.

5.3.2.7.1 Object Property Groups

Many devices currently maintain an accelerator file or database containing a subset of available metadata used by the device for its own UI. These metadata items, when defined as object properties, are much more easily and efficiently retrieved than metadata stored within the content on the device.

Devices are encouraged to identify the relative retrieval qualities of different properties by assigning them a group value in the **ObjectPropDesc** dataset, and allowing them to be retrieved based on that group value. Group codes shall be assigned in ascending order, based upon the relative retrieval speed of the property.

The expected behavior of an initiator that is taking advantage of object property groups is that it will retrieve properties in ascending order.

5.3.2.7.2 Special Values for Object Property Groups

Properties of essentially unbounded size shall be marked with a group code of 0xFFFFFFFF. This includes any property defined by a LongString or ByteArray form. It is the responsibility of the device to accurately identify these groups.

5.3.2.8 Representative Samples

It is often desirable to sample an object without acquiring the entire object, or to provide a visual representation of the object. Examples of this include adding album art to an album, representative audio clips to music files, or thumbnails to image files. These representative samples are enabled by embedding them in a property value.

There are six properties related to representative samples. The first five properties are descriptive of the representative sample, and cannot be set by the initiator. Rather, the **ObjectPropDesc** fields of those properties are provided to define the allowable values of the representative sample, and the values of the properties must be inferred from the representative sample itself.

Example:
A portable media player needs to attach representative clips of audio files to full-length songs in order to facilitate audio-only navigation of the contents of a device. To do so, the media player would identify the format(s) in which the clips are desired in the **ObjectPropDesc** dataset for the RepresentativeSampleFormat property on the audio clip formats. Further, the range of desired durations for the sample is provided in the RepresentativeSampleDuration ObjectPropDesc dataset, and the maximum file size of the sample in the RepresentativeSampleSize ObjectPropDesc dataset. This information would be sufficient for an initiator to send an appropriate clip to the device.

When an initiator is browsing that device later, it can identify the file size, duration, and format of the representative sample by retrieving those properties. The values for those properties must be populated by the responder automatically when the sample is sent to (or created on) the device.

5.3.2.9 Example of Object Properties in Use

The following table shows an overview of the object property exchange process, presented as a step-through of the dialog between an MTP Initiator and Responder.

Step	Initiator action	Parameter1	Parameter2	Parameter3	Responder action
1	**GetDeviceInfo**	0x00000000	0x00000000	0x00000000	Send DeviceInfo dataset.
2	**OpenSession**	SessionID	0x00000000	0x00000000	Create ObjectHandles and StorageIDs if necessary.
3	**GetObjectHandles**	0xFFFFFFFF	0xFFFFFFFF	0x00000000	Send

					ObjectHandle array.
4	**GetObjectPropSupported**	ObjectFormatCode	0x00000000	0x00000000	Send ObjectPropCode array.
5	**GetObjectPropDesc**	ObjectProp 1	Object Format 1	0x00000000	Send ObjectPropDesc dataset.
6	**Repeat step 5 for each ObjectProperty**	ObjectProp n	0x00000000	0x00000000	Send ObjectPropDesc dataset.
7	**GetObjectPropValue**	ObjectHandle 1	ObjectPropCode 1	0x00000000	Send value of ObjectProp 1 for ObjectHandle 1.
8	**Repeat step 7 for each ObjectHandle x ObjectPropCode**	ObjectHandle m	ObjectPropCode n	0x00000000	Send value of ObjectProp *n* for ObjectHandle *n*.
9	**CloseSession**	0x00000000	0x00000000	0x00000000	Close session.

5.3.2.10 Summary

Object properties provide a more extensible, more flexible, and higher performance method for object metadata exchange and enumeration. They allow devices to describe content separate from the binary data itself, which provides value to devices or applications which do not understand the underlying format, and allows much faster and more flexible content enumeration.

Object properties are an ongoing effort, and will continue to evolve for the lifespan of MTP. Any changes to the default set of Object Properties in MTP are strictly additive following version 1.0 of the MTP specification.

5.3.3 Object References

MTP is a file system-independent protocol, which allows more flexibility in device design. As a consequence of not being able to rely on a common and consistent addressing mechanism, MTP lacks the ability to form complex linkages between objects by embedding file names. An abstract referencing mechanism has been defined to allow arbitrary object referencing.

5.3.3.1 Object Reference Structure

The Object Reference Dataset consists of a single AUINT32, an array of 32-bit unsigned integers, which contains the Object Handle of each object which is referenced by the object to which the Object Reference Dataset in question is attached.

More specifically the dataset looks like:

Field	Size (bytes)	Format
NumElements	4	UINT32
ArrayEntry[0**]**	4	ObjectHandle
ArrayEntry[1**]**	4	ObjectHandle
…	…	…
ArrayEntry[*NumElements-1***]**	4	ObjectHandle

5.3.3.2 Setting Object References

Devices that support object-reference retrieval and are write-capable (that is, they can have objects sent to them as well as retrieved) shall also support the setting of object references.

Object references are set by passing the whole array of references for that object in the **SetObjectReference** operation, and is defined in the appropriate appendix of this specification.

This operation replaces the object references on a device with the array of object handles passed in the data phase. The object handles passed in the data phase must be maintained indefinitely, and returned as valid ObjectHandles referencing the same object in later sessions. If any of the object handles in the array passed in the data phase are invalid, the responder shall fail the operation by returning a response code of Invalid_ObjectHandle.

An object handle may be present multiple times in a single array of references. An example of this is a song that appears multiple times in a single playlist.

5.3.3.3 Retrieving Object References

If a device indicates support for object references, it must allow those object references to be retrieved by using the **GetObjectReferences** operation.

The **GetObjectReferences** operation returns an array of currently valid ObjectHandles., and is defined in the appropriate appendix of this specification.

5.3.3.4 Identifying Support for Object References

For a device to be queried for object references, it must identify support for the **GetObjectReferences** operation in the **OperationsSupported** field of its DeviceInfo dataset.

Once support for that operation has been declared, the initiator may query for and expose any and all object references.

5.3.3.5 References are Unidirectional

References are unidirectional, and one cannot determine which objects reference a given object without examining all the references on all the objects on the device. This does not prevent the device from containing this information in its internal representation of references.

5.3.3.6 The Meaning of Object References Is Contextual

Object references do not include any inherent meaning. An object either references another object, or it does not, and no additional information is contained within that definition.

In practice, the meaning will be defined by context. For example, when a media object references a DRM certificate, the DRM certification may be interpreted as being a license defining the allowed usage of that media object. If a photograph references an audio clip, that may indicate an audio annotation.

In many cases, the meaning will be unclear to the device, but the device shall maintain consistent references between objects anyway, to preserve information between connectivity sessions.

5.3.3.7 Reference Maintenance

The object handles returned must be consistent between sessions. That is, the actual values may change, but the objects they reference and the order in which those objects are listed cannot change. It is the responsibility of the device to manage those references between sessions such that they remain consistent, and to manage the removal of invalid Object References (caused by an object being deleted from a device.)

If a referenced object is deleted on the device between sessions, the device must remove all instances of that object in all other objects' references. This removal may be done lazily, when the references for an object are requested.

The unidirectional and context-dependent design of references is designed to facilitate reference maintenance on the device.

5.3.4 Basic Object Transfer

5.3.4.1 Sent Object Placement

When a new object is sent using the **SendObject** operation, the Initiator may attempt to specify the location where the new object will be placed on the device. It does this by first calling the **SendObjectInfo** operation. The parameters of the **SendObjectInfo** operation may contain the **StorageID** of the storage on the device on which the Initiator wishes to place the object and the Object Handle of the desired parent of this object (that is, the folder into which the object shall be placed).

Following the successful execution of the **SendObjectInfo** operation, the **SendObject** operation shall be the next operation to be sent to the receiving device, which will initiate the transfer of the data portion of the object. The intent is that the **SendObjectInfo** operation will provide all the required information to determine whether the object will be able to be successfully stored by the receiving device, so when the **SendObject** operation begins, it will have a high probability of success, and the receiver will know in advance how to handle the incoming object.

If the sending device provides a **StorageID** in the **SendObjectInfo** operation, the receiving device shall determine whether the specified storage and parent object can be used to locate the object on the device once it is received.

This requires that the receiving device test the following conditions in this order:

1. If the receiving device does not permit the target destination to be specified at all, it shall alert the sending device by sending a Specification_of_Destination_Unsupported failure response.
2. The **StorageID** passed shall refer to an actual storage currently present in the device.
3. The storage referred to shall have an access capability that allows write-access.
4. The storage referred to must have sufficient free space to contain the object to be sent. The size (in bytes) of the object to be sent is given in the **ObjectInfo** dataset passed in the data phase of the **SendObjectInfo** operation.
5. The parent object passed in the second parameter shall identify a valid object handle of an Association on the device.
6. Any other conditions required for the successful execution of the **SendObject** operation shall then be tested, and an appropriate response sent in the case of an expected failure.

The receiving device may choose to not support the specification of a target destination on the device for a variety of reasons. When this is indicated by the appropriate failure code, sent as a response to a **SendObjectInfo** operation that specifies a particular

StorageID/Parent Object, the Initiator shall then attempt the process again without specifying a desired destination on the device, allowing the receiving device to specify the location of the object.

Appendix A – Object Formats

A.1 Object Format Summary Table

Name	Datacode	Description
Undefined	*0x3000	Undefined object
Association	0x3001	Association (for example, a folder)
Script	0x3002	Device model-specific script
Executable	0x3003	Device model-specific binary executable
Text	0x3004	Text file
HTML	0x3005	Hypertext Markup Language file (text)
DPOF	0x3006	Digital Print Order Format file (text)
AIFF	0x3007	Audio clip
WAV	0x3008	Audio clip
MP3	0x3009	Audio clip
AVI	0x300A	Video clip
MPEG	0x300B	Video clip
ASF	0x300C	Microsoft Advanced Streaming Format (video)
defined	0x3800	Unknown image object
EXIF/JPEG	0x3801	Exchangeable File Format, JEIDA standard
TIFF/EP	0x3802	Tag Image File Format for Electronic Photography
FlashPix	0x3803	Structured Storage Image Format
BMP	0x3804	Microsoft Windows Bitmap file
CIFF	0x3805	Canon Camera Image File Format
Undefined	0x3806	Reserved
GIF	0x3807	Graphics Interchange Format
JFIF	0x3808	JPEG File Interchange Format
CD	0x3809	PhotoCD Image Pac
PICT	0x380A	Quickdraw Image Format
PNG	0x380B	Portable Network Graphics
Undefined	0x380C	Reserved
TIFF	0x380D	Tag Image File Format
TIFF/IT	0x380E	Tag Image File Format for Information Technology (graphic arts)
JP2	0x380F	JPEG2000 Baseline File Format
JPX	0x3810	JPEG2000 Extended File Format
Undefined Firmware	0xB802	
Windows Image Format	0xB881	
Undefined Audio	0xB900	
WMA	0xB901	
OGG	0xB902	
AAC	0xB903	
Audible	0xB904	
FLAC	0xB906	

Undefined Video	0xB980	
WMV	0xB981	
MP4 Container	0xB982	ISO 14496-1
MP2	0xB983	
3GP Container	0xB984	3GPP file format. Details: http://www.3gpp.org/ftp/Specs/html-info/26244.htm (page title - "Transparent end-to-end packet switched streaming service, 3GPP file format").
Undefined Collection	0xBA00	
Abstract Multimedia Album	0xBA01	
Abstract Image Album	0xBA02	
Abstract Audio Album	0xBA03	
Abstract Video Album	0xBA04	
Abstract Audio & Video Playlist	0xBA05	
Abstract Contact Group	0xBA06	
Abstract Message Folder	0xBA07	
Abstract Chaptered Production	0xBA08	
Abstract Audio Playlist	0xBA09	
Abstract Video Playlist	0xBA0A	
Abstract Mediacast	0xBA0B	For use with mediacasts; references multimedia enclosures of RSS feeds or episodic content
WPL Playlist	0xBA10	
M3U Playlist	0xBA11	
MPL Playlist	0xBA12	
ASX Playlist	0xBA13	
PLS Playlist	0xBA14	
Undefined Document	0xBA80	
Abstract Document	0xBA81	
XML Document	0xBA82	
Microsoft Word Document	0xBA83	
MHT Compiled HTML Document	0xBA84	
Microsoft Excel spreadsheet (.xls)	0xBA85	
Microsoft	0xBA86	

Powerpoint presentation (.ppt)		
Undefined Message	0xBB00	
Abstract Message	0xBB01	
Undefined Contact	0xBB80	
Abstract Contact	0xBB81	
vCard 2	0xBB82	

Appendix B – Object Properties

B.1 Object Property Summary Table

MTP Name	MTP Datacode
StorageID	0xDC01
Object Format	0xDC02
Protection Status	0xDC03
Object Size	0xDC04
Association Type	0xDC05
Association Desc	0xDC06
Object File Name	0xDC07
Date Created	0xDC08
Date Modified	0xDC09
Keywords	0xDC0A
Parent Object	0xDC0B
Allowed Folder Contents	0xDC0C
Hidden	0xDC0D
System Object	0xDC0E
Persistent Unique Object Identifier	0xDC41
SyncID	0xDC42
Property Bag	0xDC43
Name	0xDC44
Created By	0xDC45
Artist	0xDC46
Date Authored	0xDC47
Description	0xDC48
URL Reference	0xDC49
Language-Locale	0xDC4A
Copyright Information	0xDC4B
Source	0xDC4C
Origin Location	0xDC4D
Date Added	0xDC4E
Non-Consumable	0xDC4F
Corrupt/Unplayable	0xDC50
ProducerSerialNumber	0xDC51
Representative Sample Format	0xDC81
Representative Sample Size	0xDC82
Representative Sample Height	0xDC83
Representative Sample Width	0xDC84
Representative Sample Duration	0xDC85
Representative Sample Data	0xDC86
Width	0xDC87
Height	0xDC88
Duration	0xDC89
Rating	0xDC8A

Track	0xDC8B
Genre	0xDC8C
Credits	0xDC8D
Lyrics	0xDC8E
Subscription Content ID	0xDC8F
Produced By	0xDC90
Use Count	0xDC91
Skip Count	0xDC92
Last Accessed	0xDC93
Parental Rating	0xDC94
Meta Genre	0xDC95
Composer	0xDC96
Effective Rating	0xDC97
Subtitle	0xDC98
Original Release Date	0xDC99
Album Name	0xDC9A
Album Artist	0xDC9B
Mood	0xDC9C
DRM Status	0xDC9D
Sub Description	0xDC9E
Is Cropped	0xDCD1
Is Colour Corrected	0xDCD2
Image Bit Depth	0xDCD3
Fnumber	0xDCD4
Exposure Time	0xDCD5
Exposure Index	0xDCD6
Total BitRate	0xDE91
Bitrate Type	0xDE92
Sample Rate	0xDE93
Number Of Channels	0xDE94
Audio BitDepth	0xDE95
Scan Type	0xDE97
Audio WAVE Codec	0xDE99
Audio BitRate	0xDE9A
Video FourCC Codec	0xDE9B
Video BitRate	0xDE9C
Frames Per Thousand Seconds	0xDE9D
KeyFrame Distance	0xDE9E
Buffer Size	0xDE9F
Encoding Quality	0xDEA0
Encoding Profile	0xDEA1
Display Name	0xDCE0
Body Text	0xDCE1
Subject	0xDCE2
Priority	0xDCE3
Given Name	0xDD00
Middle Names	0xDD01
Family Name	0xDD02

Prefix	0xDD03
Suffix	0xDD04
Phonetic Given Name	0xDD05
Phonetic Family Name	0xDD06
Email Primary	0xDD07
Email Personal 1	0xDD08
Email Personal 2	0xDD09
Email Business 1	0xDD0A
Email Business 2	0xDD0B
Email Others	0xDD0C
Phone Number Primary	0xDD0D
Phone Number Personal	0xDD0E
Phone Number Personal 2	0xDD0F
Phone Number Business	0xDD10
Phone Number Business 2	0xDD11
Phone Number Mobile	0xDD12
Phone Number Mobile 2	0xDD13
Fax Number Primary	0xDD14
Fax Number Personal	0xDD15
Fax Number Business	0xDD16
Pager Number	0xDD17
Phone Number Others	0xDD18
Primary Web Address	0xDD19
Personal Web Address	0xDD1A
Business Web Address	0xDD1B
Instant Messenger Address	0xDD1C
Instant Messenger Address 2	0xDD1D
Instant Messenger Address 3	0xDD1E
Postal Address Personal Full	0xDD1F
Postal Address Personal Line 1	0xDD20
Postal Address Personal Line 2	0xDD21
Postal Address Personal City	0xDD22
Postal Address Personal Region	0xDD23
Postal Address Personal Postal Code	0xDD24
Postal Address Personal Country	0xDD25
Postal Address Business Full	0xDD26
Postal Address Business Line 1	0xDD27
Postal Address Business Line 2	0xDD28
Postal Address Business City	0xDD29
Postal Address Business Region	0xDD2A
Postal Address Business Postal Code	0xDD2B
Postal Address Business Country	0xDD2C
Postal Address Other Full	0xDD2D
Postal Address Other Line 1	0xDD2E
Postal Address Other Line 2	0xDD2F
Postal Address Other City	0xDD30
Postal Address Other Region	0xDD31
Postal Address Other Postal Code	0xDD32

Postal Address Other Country	0xDD33
Organization Name	0xDD34
Phonetic Organization Name	0xDD35
Role	0xDD36
Birthdate	0xDD37
Message To	0xDD40
Message CC	0xDD41
Message BCC	0xDD42
Message Read	0xDD43
Message Received Time	0xDD44
Message Sender	0xDD45
Activity Begin Time	0xDD50
Activity End Time	0xDD51
Activity Location	0xDD52
Activity Required Attendees	0xDD54
Activity Optional Attendees	0xDD55
Activity Resources	0xDD56
Activity Accepted	0xDD57
Activity Tentative	0xDD58
Activity Declined	0xDD59
Activity Reminder Time	0xDD5A
Activity Owner	0xDD5B
Activity Status	0xDD5C
Owner	0xDD5D
Editor	0xDD5E
Webmaster	0xDD5F
URL Source	0xDD60
URL Destination	0xDD61
Time Bookmark	0xDD62
Object Bookmark	0xDD63
Byte Bookmark	0xDD64
Last Build Date	0xDD70
Time to Live	0xDD71
Media GUID	0xDD72

B.2 Object Property Descriptions

B.2.1 StorageID

This property represents the storage on which this object exists. If the removal or formatting of a storage would cause this object to be corrupted or removed, this property shall identify the storage upon which this object depends, even if it does not have a binary component. This property must contain the same value as the first field of the ObjectInfo dataset.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	4	UINT32	0xDC01
Datatype	2	4	UINT32	0x0006 (UINT32)
Get/Set	3	1	UINT8	0x00 (GET)
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.2 Object Format

The object format code describes this object. For more information about object format codes see section 4, "Object Formats".
This property must contain the same value as the second field of the ObjectInfo dataset.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC02
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	0x00 (GET)
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.3 Protection Status

This property identifies the write-protection status of the binary component of the data object. Read-only protection may modified with the SetObjectProtection operation if

allowed by the device. Read-only protection may not be modified by mechanisms used to set Object Properties.

The allowed values for this property for a device shall be enumerated in ObjectPropDesc dataset and are defined as follows:

Value	Description
0x0000	No Protection This object has no protection, it may be modified/deleted arbitrarily, and its properties may be modified freely.
0x0001	Read-only This object cannot be deleted or modified, and all properties of this object cannot be modified by the initiator. (Properties may be modified by the device which contains the object however.)
0x8002	Read-only data This object's binary component cannot be deleted or modified, however any object properties may be modified if allowed by the object property constraints.
0x8003	Non-transferrable data This object's properties may be read and modified, and it may be moved or deleted on the device, but this object's binary data may not be retrived from the device using a GetObject operation.
0x0002-0x7FFF	Reserved for PTP
0x8004-0x8BFF	Reserved for MTP
0x8C00-0xFFFF	MTP Vendor Extension Range

This property may be indirectly set using the legacy SetObjectProtection operation, as well as the usual mechanisms used to set Object Properties.

Field name	**Field order**	**Size (bytes)**	**Datatype**	**Value**
PropertyCode	1	2	UINT16	0xDC03
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	0x00 (GET)
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.4 Object Size

This property provides the size of the binary component of the object, in bytes. It must be identical to the value contained in the fourth field of the ObjectInfo dataset (ObjectCompressedSize), unless it is greater than the maximum size of that field. When applied to an object that does not have a defined object size (such as an abstract object without a data component or an object created on demand) this property shall contain 0x00000000.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC04
Datatype	2	2	UINT16	0x0008 (UINT64)
Get/Set	3	1	UINT8	0x00 (GET)
DefaultValue	4			0x0000000000000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.5 Association Type

This property defines the association type of the object. It only applies to objects that have an object format value of 0x3001 (Association). In general, MTP only requires the use of 0x0001. If this object is not an association, the property shall have a value of 0x0000.

The FORM fields of this property's Object Property Description dataset shall contain an enumeration of allowed values for this property, and may include:

0x0000 Undefined
This value indicates that this association has no known folder type.

0x0001 Generic Folder
This property defines the association type of the object. It only applies to objects that have an object format value of 0x3001 (Association). In general, MTP requires only the use of 0x0001. If this object is not an association, the property shall have a value of 0x0000.

This property must contain the same value as the field 13 of the ObjectInfo dataset.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC05
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.6 Association Desc

This property provides additional information about Association objects, and its meaning is different for each different Association type.

The most common use of this field is to identify an association as an MTP-compliant hierarchical folder by setting the value to 0x00000001. This indicates that this Association contains references to each object "contained" in it.

If this object is not an Association, this property shall either not be supported for the object, or the value of this property must be set to 0x00000000.

This property must contain the same value as the field 14 of the ObjectInfo dataset.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC06
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.7 Object File Name

This property identifies the file name of the object. It does not have to be human-readable, and may or may not reference the actual file name of the object on the device. The value of this property shall not contain path information.

The FORM fields of the object property description dataset for this property are used to further define the file names which may be used to identify files on the device. If this value is a null string, it is assumed that any string may be placed in this property.

Some examples of common regular expressions that might be used for a device include the following (no guarantee is made for correctness; these examples are provided as-is only):
Strings containing only alphanumeric characters:
[a-zA-Z0-9]*
Strings not containing the "\" or "/" characters:
[^\\/]*
FAT12/FAT16 compatible file names:
[a-zA-Z!#\$%&'\(\)\-0-9@\^_\`\{\}\~][a-zA-Z!#\$%&'\(\)\-0-9@\^_\`\{\}\~]{0,7}\.[[a-zA-Z!#\$%&'\(\)\-0-9@\^_\`\{\}\~][a-zA-Z!#\$%&'\(\)\-0-9@\^_\`\{\}\~]{0,2}]?

The value of this property must be identical to the value of field 16 of the ObjectInfo dataset.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC07
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None 0x05 RegularExpression form

B.2.8 Date Created

This property contains the date and time when the object was created. This does not refer to the date when the object was originally authored, but rather to the date when this particular binary object was first created (for example, an audio file generated from a CD would contain the date and time when the file was generated, not when the work was performed or written).

The value of this property must be identical to the value of the seventeenth field of the ObjectInfo dataset.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC08
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x03 DateTime form

B.2.9 Date Modified

This property contains the date and time when the object was last altered. If this property is supported, it shall be updated to the current time whenever an object's binary component, properties, or references are updated.

The value of this property must be identical to the value of the eighteenth field of the ObjectInfo dataset.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC09
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x03 DateTime form

B.2.10 Keywords

This property defines a string containing a list of keywords associated with the object, each separated by a ‘ ‘ character.

The value of this property must be identical to the value of field 19 of the ObjectInfo dataset.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC0A
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.11 Parent Object

If this object exists in a hierarchy, this property will contain the object handle of the parent of this object. Only objects of type Association may be identified in this field. If this object is in the root or the device does not support associations, this field shall be set to 0x00000000.

Devices cannot be moved by updating this property. They must be moved using the MoveObject operation.

The value of this property must be identical to the value of field 12 of the ObjectInfo dataset.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC0B
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	0x00 (GET)
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.12 Allowed Folder Contents

This property may only be supported for objects with an object format type of Association and which represent Generic Folders.

In many cases, a folder object may have a restriction on the contents allowed to be placed in that folder. If this is the case, this property shall be used to indicate this restriction to the Initiator. This property contains an array of datacodes representing the content types which may be placed immediately within the folder to which this property is attached. This does not restrict the entire hierarchy of objects contained by this folder, only the objects which may exist immediately within this folder object. If nested folders may be placed in this folder, that shall be indicated by including the Object Format Type for Association (0x3001) in the array for this folder. If the content of a folders contained within this folder must be restricted as well, that folder shall contain its own restriction and represent that through this property on the nested folder.

If there is no restriction on the contents of a folder (beyond the overall device restriction as indicated in the DeviceInfo dataset), the value of property shall be an empty array.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC0C
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	0x00 (GET)
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.13 Hidden

This property identifies whether an object is intended to be displayed to a user, or whether it is intended to be hidden and used only by applications. An object which is identified as hidden should only be hidden from users browsing the contents of a device through a friendly user interface, and only at the discretion of the displaying application. Objects which are hidden should never be hidden from other applications.

This property is not to be used to hide items from an initiator. If a responder does not wish to expose an object to applications on an initiator, it must not enumerate the object at all through MTP.

Valid values of this property are defined as follows:
0x0000 This object is intended to be visible to users.
0x0001 This object is intended to be hidden from non-technical users.
All other values where Bit 15 is 0 are reserved for MTP
All other values where Bit 15 is 1 are the MTP Vendor Extension range

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC0D
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.14 System Object

This property identifies whether an object is a system file, and is required for property functioning of a device (or a core application on a device). An object which is identified as a system file may be exposed or manipulated differently by applications on a PC, but no explicit restrictions are placed on the use of a system file by MTP. It is suggested that system files be excluded from automated synchronization processes and hidden from non-technical users.

Valid values of this property are defined as follows:
0x0000 This object is intended to be visible to users.
0x0001 This object is intended to be hidden from non-technical users.
All other values where Bit 15 is 0 are reserved for MTP
All other values where Bit 15 is 1 are the MTP Vendor Extension range

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC0E
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.15 Persistent Unique Object Identifier

This property is set by the responder and cannot be altered by the initiator. As long as this object is present on the device, it must contain the same persistent unique object identifier, and it shall be the only object with that identifier. Every effort must be made to ensure that no persistent unique object identifier value is ever re-used on the device.

The primary purpose of this property is to ensure that an initiator that repeatedly establishes sessions with the device can know which objects it has previously enumerated or acquired.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC41
Datatype	2	2	UINT16	0x000A (UINT128)
Get/Set	3	1	UINT8	0x00 (GET)
DefaultValue	4			0x0…0
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.16 SyncID

This property is set by the initiator, and shall not be altered by the responder.
This property is opaque to the responder and is intended to allow an initiator to retain state between sessions without retaining a catalogue of connected device content. (This is particularly useful in sync-server scenarios – remembering that there should be one of these for each distinct Intiator.)

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC42
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.17 Property Bag

In many cases, an initiator wants to save an object property associated with an object that is not supported by the responder. In those cases, the initiator may update the Property_Bag value with that information.

The Property_Bag is a property which contains a collection of properties in an XML document. It is not expected to be understood by the responder, but the contents shall be preserved and returned to an initiator upon request.

The schema for this XML file is outside of the scope of this document.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC43
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.18 Name

This property contains the name of the object. In many cases this will overlap with other properties, such as File Name, and can be seen as a consistently available, unique, human-readable identifier.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC44
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.19 Created By

This property identifies the application, user, or organization that originally created the binary object to which this property applies. This property is not intended to identify the person who created the intellectual property contained within this object; that information is intended to be contained in the Artist property (0xDD06).

If this object was created on the device, this field is intended to contain some combination of the "Manufacturer" and "Model" fields of the DeviceInfo dataset, or alternatively the value contained in the "DeviceFriendlyName" device property.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC45
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.20 Artist

This property identifies the person or people who originally created this object. In cases where the object to which this property applies is an artistic work, this property identifies the creators of that work. In cases where the object to which this property applies is a document, this property identifies the author. Etc.

This property differs from the CreatedBy property in that it always identifies a person. This property and the CreatedBy property may often contain the same value.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC46
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.21 Date Authored

This property contains the date and time when the content in this object was originally created. In cases when the object to which this property applies is an artistic work, this property implies the date when that work was created.

The Date Authored and Date Created properties may contain the same value.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC47
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x03 DateTime form

B.2.22 Description

This property contains a human-readable description of the object.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC48
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.23 URL Reference

This property contains a reference to a URL for this object. The location linked to is not defined by this protocol.

The FORM fields of this property may contain a Regular Expression limiting this field to valid HTML addresses, as follows:
((http:)?/{0,2}([^/\?#\[\];:&=\+\$,]*\.)+([^/\?#\[\];:&=\+\$,]{2,3}))(/[^<>])*
or it may contain a null string to indicate that the value is not used or validated.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC49
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.24 Language-Locale

This property identifies the language of this object. If multiple languages are contained in this object, it shall identify the primary language (if any).

This property may contain either a language code, as defined in ISO-639, such as:
"en"
"ja"

It may also contain a language-country code, which consists of a language code of two or three characters as defined in the ISO-639 standard, followed by a hyphen, then followed by a country code as defined in ISO-3166, such as:
"en-US"
"ja-JP"

The FORM fields of this property may contain a Regular Expression limiting this field to valid Language-Locales, such as:
[a-zA-Z]{2,3}(-[a-zA-Z]{2})?
or it may contain a null string to indicate that the value is not used or validated.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC4A
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.25 Copyright Information

This property contains the copyright information for the object.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC4B
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.26 Source

This property contains the source of this object. In general, this is not intended to identify an individual or organization. For audio files, this is intended to contain the collection from which the work was retrieved (generally the album name).

This property may overlap with the Artist and CreatedBy properties for some object formats, and is intended to be implemented only if those two properties are insufficient or inapplicable.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC4C
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.27 Origin Location

This property contains the origin location for this object.

There are many ways to identify a location. The method used shall be represented by using a regular expression in the FORM field of the Object Property Description dataset, which must describe the desired location format. However, given the generality of FORM regular expressions, this spec does not intend that Initiators and Reponders will be able to determine the precise method in use by examining the FORM field alone.

Some example regular expressions:
ISO-3166 country code:
[A-Z]{2}
WGS-84 GPS location (String):
[NS] [1-9][0-9]{0,2}d [1-9][0-9]?,[0-9]{1,9}m [EW] [1-9][0-9]{0,2}d [1-9][0-9]?,[0-9]{1,9}m
WGS-84 GPS location (decimal):
[1-9][0-9]{1,2}\.[0-9]{1,9} [1-9][0-9]{1,2}\.[0-9]{1,9}

If the FORM field contains a null string, it indicates that any human-readable string may be placed in this field (such as City/Country name).

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC4D
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.28 Date Added

This property identifies the time and date when this object was added to the device, as identified by the internal clock on the device.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC4E
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	0x00 (GET)
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x03 DateTime form

B.2.29 Non-Consumable

This property determines whether or not this object is intended to be consumed by the device, or whether it has been placed on the device just for storage. If this property is not present, all data is assumed to be intended for consumption.

If this property is supported, it must be supported for all objects on the device.
The allowed values for this property are identified in the FORM fields of the Object Property Description dataset, and must contain the values 0x00 and 0x01. A value of 0x00 indicates that the object is intended for consumption, and a value of 0x01 indicates that it shall simply be stored. If a vendor-specific extension is defined which extends the allowed values of this property, it shall use only values with a most significant bit of 1.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC4F
Datatype	2	2	UINT16	0x0002 (UINT8)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.30 Corrupt/Unplayable

This property identifies an object on the device that should be able to be understood, but for some reason, cannot be played. This should not be applied to objects which are non-consumable. This property may only be set by the device as a result of failing to consume the data portion of an object.

The allowed values for this property are 0x00 and 0x01. A value of 0x01 indicates that the object is corrupt or unplayable. If a vendor-specific extension is defined which extends the allowed values of this property, it shall use only values with a most significant bit of 1.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC50
Datatype	2	2	UINT16	0x0002 (UINT8)
Get/Set	3	1	UINT8	0x00 (GET)
DefaultValue	4			0x00
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.31 ProducerSerialNumber

This property identifies the unique serial number of the device which originally created the binary object to which this property applies.

If this object was created by this device, this field shall contain the contents of the Serial Number field of the DeviceInfo datase.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC51
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.32 Representative Sample Format

This property identifies the object format of the representative sample for the object, using an MTP Object Format Type datacode.

The FORM fields in the Object Property Description dataset shall indicate the supported Representative Sample format types for a given object format type in an enumeration.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC81
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.33 Representative Sample Size

This property identifies the size in bytes of the representative sample for this object.
The FORM fields in the Object Property Description dataset shall indicate the supported sizes of Representative Samples of this object format type in a Range FORM.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC82
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x01 Range form

B.2.34 Representative Sample Height

This property identifies the height of the representative sample for the object in pixels. The FORM fields in the Object Property Description dataset shall indicate the supported heights of Representative Samples of this object format type in a Range FORM.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC83
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x01 Range form

B.2.35 Representative Sample Width

This property identifies the width of the representative sample for the object in pixels.

The FORM fields in the Object Property Description dataset shall indicate the supported widths of Representative Samples of this object format type in a Range FORM.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC84
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x01 Range form

B.2.36 Representative Sample Duration

This property identifies the duration of the representative sample for the object in milliseconds.

The FORM fields in the Object Property Description dataset shall indicate the supported durations of Representative Samples of this object format type in a Range FORM.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC85
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x01 Range form

B.2.37 Representative Sample Data

This property contains a representative sample of the object to which it applies.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC86
Datatype	2	2	UINT16	0x4002 (AUINT8)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x06 ByteArray form

B.2.38 Width

This property identifies the width of an object in pixels.

The FORM fields in the Object Property Description dataset shall indicate the supported widths of objects of this format type in a Range FORM.

If this property is read-only, it must be calculated by the device based on the object when requested. If this property is read-write, the device may return the default value (0x00000000) when it has not yet extracted the correct value from the object, and may allow the value to be set in by the initiator (though it shall correct the value, when possible, if the value set by the initiator is incorrect).

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC87
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x01 Range form

B.2.39 Height

This property identifies the height of an object in pixels.

The FORM fields in the Object Property Description dataset shall indicate the supported heights of objects of this format type in a Range FORM.

If this property is read-only, it must be calculated by the device based on the object when requested. If this property is read-write, the device may return the default value (0x00000000) when it has not yet extracted the correct value from the object, and may allow the value to be set in by the initiator (though it shall correct the value, when possible, if the value set by the initiator is incorrect).

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC88
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x01 Range form

B.2.40 Duration

This property identifies the duration of an object in milliseconds.

The FORM fields in the Object Property Description dataset shall indicate the supported durations of objects of this format type in a Range FORMLESS.

If this property is read-only, it must be calculated by the device based on the object when requested. If this property is read-write, the device may return the default value (0x00000000) when it has not yet extracted the correct value from the object, and may allow the value to be set in by the initiator (though it shall correct the value, when possible, if the value set by the initiator is incorrect).

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC89
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 Range formless

B.2.41 Rating

This property contains the value of rating for the object, as set by a user. This represents a rating of how much this object is appreciated (such as a "star" rating from 1 to 5 stars), and does not identify a maturity rating (such as R or PG-13).

The user rating is always exchanged as a value from 1 to 100. If the user rating has not been set, it shall have a value of 0. The FORM fields of the Object Property Description dataset for this property shall identify a range from 0 to 100 with a step-size of 1.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC8A
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x01 Range form

B.2.42 Track

This property identifies the track on which this object is found on its distribution media.

It primarily applies to objects which are also distributed on optical media, such as CDs and DVDs, and generally is set by the initiator. A value of 0x0000 (default value) indicates that it is not in use, so track numbers shall be 1-based.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC8B
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.43 Genre

This property identifies the genre of this object. This genre may be any human-readable genre-describing string, and generally must be set by the initiator.

The device shall indicate the list of supported genres in an enumeration FORM in the Object Property Description dataset. If the enumeration contains a null string (or the device does not contain an enumeration FORM in the Object Property Description dataset), this indicates that any human-readable string identifying genre may be placed in this property.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC8C
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.44 Credits

This property identifies credits for the object. The format of these credits shall be simple text and the value is generally set by the initiator.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC8D
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.45 Lyrics

This property contains the lyrics or script of the property. The format of these lyrics shall be simple text and the value is generally set by the initiator.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC8E
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.46 Subscription Content ID

This property provides additional information to identify a piece of content relative to an online subscription service. It is generally set by the initiator and is a specific format.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If no specific subscription service is supported, or there is no constraint on the subscription identifier value, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC8F
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.47 Produced By

This property identifies the person or organization that produced this work. It primarily applies to audio and video content, and is generally set by the initiator.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC90
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.48 Use Count

This property identifies the number of times this object has been played or viewed.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC91
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.49 Skip Count

This property identifies the number of times this object was set up to be played, but manually skipped by the user. It is intended to be used to infer a rating for an object or to modify content update rules.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC92
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.50 Last Accessed

This property contains the date and time when this object was last viewed, accessed, or otherwise used, relative to a device's onboard clock. It may be set by the initiator when an object exists both on the device and on the initiator and consistency is desired, and it shall be updated on the device.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC93
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x03 DateTime form

B.2.51 Parental Rating

This property identifies the parental rating assigned to this object. The purpose of this property is to identify objects that may not be appropriate to be viewed or heard by minors.

The contents of this field are intended to be human-readable, but may be further defined in the future to facilitate machine interpretation.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC94
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.52 Meta Genre

This property further qualifies a piece of media in a contextual way.

The device shall identify supported metagenre values in an Enumeration in the FORM fields of the Object Property Description dataset.

The following values are defined:
0x0000 Not Used
0x0001 Generic music audio file
0x0011 Generic non-music audio file
0x0012 Spoken-Word Audio Book Files
0x0013 Spoken-Word Files (non-Audio book)
0x0014 Spoken-Word News
0x0015 Spoken-Word Talk Shows
0x0021 Generic video file
0x0022 News video file
0x0023 Music video file
0x0024 Home video file
0x0025 Feature Film video file
0x0026 Television Show video file
0x0027 Training/Educational video file
0x0028 Photo montage video file
0x0030 Generic non-audio, non-video
0x0040 Audio Mediacast
0x0041 Video Mediacast
0x0042 Mixed-media Mediacast
All other values where Bit 15 is 0 are reserved for MTP
All other values where Bit 15 is 1 are the MTP Vendor Extension range

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC95
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.53 Composer

This property identifies the composer when the composer is not the artist who performed it. It applies primarily to musical works, but can be applied to any created performance. This property may often overlap with the "created by" and "artist" properties.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC96
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.54 Effective Rating

This property contains an assigned rating for an object. This rating is not set by the user, but is generated based upon usage statistics (such as usecount or skipcount), or set by an external authority. This represents a rating of how much this object is appreciated.

The effective rating is always exchanged as a value from 1 to 100. If the effective rating has not been set, it shall have a value of 0. The FORM fields of the Object Property Description dataset for this property shall identify a range from 0 to 100 with a step-size of 1.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC97
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.55 Subtitle

This property further qualifies the title when the title is ambiguous or general, such as when the title describes a series and the subtitle represents an episode title, or when the title describes a larger work and the subtitle describes an act or contained performance.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC98
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.56 Original Release Date

This property contains the date and time when this object was originally released. It applies to video works broadcast on television and audio works broadcast on radio, as well as work released on physical media.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC99
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x03 DateTime form

B.2.57 Album Name

If an object has been distributed as part of an album, the album which contained it shall be identified by this property. If this object is contained in an Album on the device, this property shall be identical to the "Name" property on the album object which contains the object to which this property is applied.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC9A
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.58 Album Artist

When an object was distributed in an album, the album artist (or artists) may differ from the artist who created the work in question. If this object is applied to an Album on the device, this property shall be identical to the "Artist" property on the album object which contains the object to which this property is applied.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC9B
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.59 Mood

This property describes the "Mood" of the object. This differs from Genre and Metagenre in implementation, and allows greater specificity and usage. The device shall indicate which moods it supports in the enumeration form fields in the Object Property Description dataset for this property. If the device supports arbitrary values, it shall contain a null string in the enumeration.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC9C
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.60 DRM Status

This property identifies the digital rights manangement (DRM) status of the object, and exists independently of any particular DRM scheme implemented in an MTP extension set. The purpose of this property is to expose in a DRM-agnostic way whether a data object will be playable after being transferred to a different device.

Valid values of this property are defined as follows:
0x0000 This object has no DRM protection.
0x0001 This object has DRM protection.
All other values where Bit 15 is 0 are reserved for MTP
All other values where Bit 15 is 1 are the MTP Vendor Extension range

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC9D
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.61 Sub Description

This property further qualifies the description when the main description is ambiguous or general, such as when the description describes a series and the subdescription represents an episode synopsis, or when the description describes a larger work and the subdescription describes an act or contained performance.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDC9E
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.62 Is Cropped

This property generally applies only to image formats. It signals whether the file has been cropped. The purpose of this property is to prevent multiple devices from automatically cropping the same image in post-processing.

Valid values of this property are defined as follows:
0x0000 This image has not been cropped.
0x0001 This image has been cropped.
0x0002 This image has not been cropped, and shall not be cropped.
All other values where Bit 15 is 0 Reserved for MTP
All other values where Bit 15 is 1 MTP Vendor Extension range

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDCD1
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.63 Is Colour Corrected

This property generally only applies to image formats. It indicates whether the file has been color corrected. The purpose of this property is to prevent multiple devices from automatically colour correcting the same image in post-processing.

Valid values of this property are defined as follows:
0x0000 This image has not been color corrected.
0x0001 This image has been color corrected.
0x0002 This image has not been color corrected, and shall not be color corrected.
All other values where Bit 15 is 0 Reserved for MTP
All other values where Bit 15 is 1 MTP Vendor Extension range

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDCD2
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.64 Image Bit Depth

This property generally only applies to image formats, and identifies the identifies the bit depth of an image.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDCD3
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x01 Range-Form 0x02 Enumeration-Form

B.2.65 Fnumber

This object property identifies the aperture setting of the lens at the time when this object was captured. This property contains the F-number scaled by 100.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDCD4
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.66 Exposure Time

This object property identifies the shutter speed of the device in seconds, scaled by 10,000, at the time when this object was captured.

Typically, a device supports discrete enumerated values for this property, which shall be identified in an enumeration FORM, but continuous control over a range is possible.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDCD5
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x01 Range-Form 0x02 Enumeration-Form

B.2.67 Exposure Index

This object property identifies the film speed emulated on the digital camera at the time when this object was captured. The settings of this property correspond to the ISO designations (ASA/DIN).

Typically, a device supports discrete enumerated values for this property, which are intended to be identified in an enumeration FORM, but continuous control over a range is possible.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDCD6
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x01 Range-Form 0x02 Enumeration-Form

B.2.68 Total BitRate

This object property specifies the total number of bits required to store one second of audio, video, or combined audio and video content. For an audio file, this property has the same value as the AudioBitRate object property. For a video file, this property has the same value as the VideoBitRate object property. For a file that combines audio and video, the value of this property is equal to the sum of the AudioBitRate and VideoBitRate property values.

The FORM fields in the Object Property Description dataset are intended to identify either a range or an enumeration of allowed values for the total bitrate, and indicate that any object with a total bitrate outside of this range will not be playable.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDE91
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x01 Range-Form 0x02 Enumeration-Form

B.2.69 Bitrate Type

This property further qualifies the bit rate of an audio/video object by identifying the type of bitrate which is being measured.

The device shall identify supported values in an Enumeration in the FORM fields of the Object Property Description dataset.

Valid values of this property are defined as follows:
0x0000 Unused
0x0001 Discrete
0x0002 Variable
0x0003 Free
All other values where Bit 15 is 0 are reserved for MTP
All other values where Bit 15 is 1 are the MTP Vendor Extension range

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDE92
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.70 Sample Rate

This property applies primarily to audio and video data, and describes the number of times an analogue media selection was sampled per second during encoding.

The FORM fields in the Object Property Description dataset shall identify either a range or an enumeration of allowed values for the total samplerate, and indicate that any object with a samplerate outside of this range will not be playable.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDE93
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x01 Range-Form 0x02 Enumeration-Form

B.2.71 Number Of Channels

This property identifies the number of channels of audio contained in an audio object.

The FORM fields in the Object Property Description dataset shall identify an enumeration of allowed values for the number of channels, and indicate that any object with a number of channels outside of this enumeration will not be playable.

Valid values of this property are defined as follows:
0x0000 Unused
0x0001 Mono (1 channel)
0x0002 Stereo (2 channels)
0x0003 2.1 channels
0x0004 3 channels
0x0005 3.1 channels
0x0006 4 channels
0x0007 4.1 channels
0x0008 5 channels
0x0009 5.1 channels
0x000A 6 channels
0x000B 6.1 channels
0x000C 7 channels
0x000D 7.1 channels
0x000E 8 channels
0x000F 8.1 channels
0x0010 9 channels
0x0011 9.1 channels
0x0012 5.2 channels
0x0013 6.2 channels
0x0014 7.2 channels
0x0015 8.2 channels
0x0016 9.2 channels
All other values where Bit 15 is 0 are reserved for MTP
All other values where Bit 15 is 1 are the MTP Vendor Extension range

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDE94
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined

FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.72 Audio BitDepth

This property identifies the audio bit depth of an audio object.

The FORM fields in the Object Property Description dataset shall identify an enumeration of allowed values for the audio bit depth, and indicate that any object with an audio bit depth outside of this range will not be playable. If the enumeration contains an entry with a value of 0x00000000, it indicates that any bit depth will be accepted.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDE95
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.73 Scan Type

This property identifies scan type used in this video object.

Valid values of this property are defined as follows:

0x0000 Unused
Unused

0x0001 Progressive
Indicates progressive frames.

0x0002 FieldInterleavedUpperFirst
Line interleaved Frames with the Upper field on the first line.

0x0003 FieldInterleavedLowerFirst
Line interleaved frames with the Lower field on the first line.

0x0004 FieldSingleUpperFirst
Fields are sent as independent samples. The field is indicated (on a per sample basis)

0x0005 FieldSingleLowerFirst
Fields are sent as independent samples. The field is indicated (on a per sample basis

0x0006 MixedInterlace
The content may contain a mix of interlaced modes

0x0007 MixedInterlaceAndProgressive
The content may contain a mix of interlaced and progressive modes.

All other values where Bit 15 is 0 are reserved for MTP
All other values where Bit 15 is 1 are the MTP Vendor Extension range

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDE97
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.74 Audio WAVE Codec

This property provides the WAVE Codec Tag for audio codecs as described in: http://msdn2.microsoft.com/en-us/library/ms867195.aspx (page title: "Registered FOURCC Codes and WAVE Formats").

The FORM fields in the Object Property Description dataset shall identify an enumeration of WAVE Codec Tags of allowed codecs for objects of this Object Format Type, and indicate that any object encoded with a codc not in this codec enumeration will not be playable. If the enumeration contains an entry with a value of 0x00000000, it indicates that any codec will be accepted for this Object Format.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDE99
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.75 Audio BitRate

This property applies to audio data (potentially contained in a video file), and describes the total number of bits that one second of content will consume.

The FORM fields in the Object Property Description dataset shall identify either a range or an enumeration of allowed values for the total audio bitrate, and indicate that any object with a total audio bitrate outside of this range will not be playable.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDE9A
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x01 Range-Form 0x02 Enumeration-Form

B.2.76 Video FourCC Codec

This property provides the FourCC Codec Tag for video codecs as described in: http://msdn2.microsoft.com/en-us/library/aa904731.aspx (page title: "Registered FOURCC Codes and WAVE Formats").

The FORM fields in the Object Property Description dataset shall identify an enumeration of FourCC Codec Tags of allowed codecs for objects of this Object Format Type, and indicate that any object encoded with a codec not in this codec enumeration will not be playable. If the enumeration contains an entry with a value of 0x00000000, it indicates that any codec will be accepted for this Object Format.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDE9B
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.77 Video BitRate

This property applies to video data, and describes the total number of bits that one second of content will consume.

The FORM fields in the Object Property Description dataset shall identify either a range or an enumeration of allowed values for the total video bitrate, and indicate that any object with a total video bitrate outside of this range will not be playable.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDE9C
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x01 Range-Form 0x02 Enumeration-Form

B.2.78 Frames Per Thousand Seconds

This property identifies the number of frames in one second of video content. It is represented in thousandths of a frame, so a value of 29.97 frames per second (such as NTSC) would be represented by a value of 29970.

The FORM fields in the Object Property Description dataset shall identify the different values for this property supported by the device for objects of this Object Format Type, and indicate that any object encoded with a number of frames per thousand seconds not identified by the FORM fields i this ataset will not be playable. The FORM fields of the dataset may be either an enumeration or a range. If the enumeration contains an entry with a value of 0x00000000, it indicates that any video content will be accepted regardless of the number of frames per thousand seconds.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDE9D
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x01 Range-Form 0x02 Enumeration-Form

B.2.79 KeyFrame Distance

The Key Frame Distance determines the maximum spacing between key frames (I-frames), and is specified in milliseconds.

The FORM fields in the Object Property Description dataset shall identify a range of allowed values for the distance between key frames, and indicate that any object with a spacing outside of this range will not be playable.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDE9E
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x01 Range form

B.2.80 Buffer Size

Buffer size is used to indicate the amount of buffering required by this object in order to decode it.

The FORM fields in the Object Property Description dataset shall identify a range of buffer sizes supported by the device, and indicate that any object requiring a buffer size to decode outside of this range will not be playable.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDE9F
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x01 Range form

B.2.81 Encoding Quality

Media codecs often include a "quality" parameter, which is generally an integer value. The meaning of this property is dependent upon the codec of the media object to which it is applied.

The FORM fields in the Object Property Description dataset shall identify a range of encoding qualities supported by the device, and indicate that any object encoded with a quality outside of this range will not be playable.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDEA0
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x01 Range form

B.2.82 Encoding Profile

Media codecs may be encoded in accordance with a profile, which defines a particular encoding algorithm or optimization process. The meaning of this property is dependent upon the codec of the media object to which it is applied, and the device should enumerate all valid values in a standard computer-readable format for each Object Format supporting this property.

This property shall be used if the Profile information is required to decode the bitstream of the encoded media. If "profile" is only used to identify a collection of encoding parameters, those should instead be identified by the other appropriate MTP properties, and this property would then become redundant.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDEA1
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.83 Display Name

This property identifies the display name for an object. The display name is the human-readable string presented to a user which identifies an object in the on-device UI.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of the display name for a device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDCE0
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.84 Body Text

This property contains an abstract copy of the body text of an object. The meaning of this property depends on the object format type of the object to which it is attached, but generally represents a plain text or html representation of a primary text field of a binary object. If this property contains HTML-formatted content, it shall have <HTML> at the beginning of the property to indicate this.

Examples of this property include:
E-Mail body text
Document body text
Etc.

The length of this property may be limited by the MaxLength field in the LongString FORM fields of the Object Property Description dataset describing this property.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDCE1
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.85 Subject

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDCE2
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.86 Priority

This property identifies the priority of an object. Object Priority is usually attached to an e-mail or calendar/task item, but may be applied to any object for which the meaning can be inferred by the device.

This property contains an integer which represents the identified priority of the object to which this property is attached. The priority represented by the value of this property is in descending ascending order, with a value of 1 being the highest possible priority, and larger values becoming progressively lower priority as the priority value increases. A value of '0' in the priority property of an object indicates that the priority is not in use or has not been assigned, and is not a valid priority value.

All valid priorities for an object format type on a device shall be identified in an enumeration form in the Object Property Description dataset for this property.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDCE3
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.87 Given Name

This property identifies the given name for a contact. In western culture, the given name is the first name of the contact.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of the given name for a device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD00
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.88 Middle Names

This property identifies the middle names for a contact. If the contact has more than one middle name, they shall all be in this property, separated by ' ' characters.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of the middle names for a device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD01
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.89 Family Name

This property identifies the family name for a contact. In western culture, this is often referred to as the "last" name.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of the family name for a device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD02
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.90 Prefix

This property identifies the prefix for a contact. In English, this includes prefixes such as Mr., Ms., Rev., etc.

This property shall contain an Enumeration FORM in its Object Property Description dataset which identifies the allowed values for this field. If there is no predefined set of prefixes used by this device, the enumeration shall include an entry of a null string, indicatng that any value may be used.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD03
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.91 Suffix

This property identifies the suffix for a contact. In English, this includes suffixes such as MD, Sr., PhD., etc.

This property shall contain an Enumeration FORM in its Object Property Description dataset which identifies the allowed values for this field. If there is no predefined set of suffixes used by this device, the enumeration shall include an entry of a null string, indicatng that any value may be used.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD04
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.92 Phonetic Given Name

This property identifies the phonetic given name for a contact. This corresponds to the yomi reading of a Japanese name, or Pinyin reading of a Chinese name.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of the phonetic given name for a device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD05
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.93 Phonetic Family Name

This property identifies the phonetic family name for a contact. This corresponds to the yomi reading of a Japanese name, or Pinyin reading of a Chinese name.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of the phonetic family name for a device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD06
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.94 Email Primary

This property contains the primary e-mail address for a contact. If multiple e-mail address properties are supported for a contact, this property may contain the same contents as a more specific property.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of an e-mail address for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

An example of a regular expression which may partially validate e-mail addresses is:
^[\w-\.]{1,}\@([\da-zA-Z-]{1,}\.){1,}[\da-zA-Z-]{2,3}$

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD07
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.95 Email Personal 1

This property contains the primary personal e-mail address for a contact.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of an e-mail address for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

An example of a regular expression which may partially validate e-mail addresses is: ^[\w-\.]{1,}\@([\da-zA-Z-]{1,}\.){1,}[\da-zA-Z-]{2,3}$

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD08
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.96 Email Personal 2

This property contains a secondary personal e-mail address for a contact.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of an e-mail address for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

An example of a regular expression which may partially validate e-mail addresses is: ^[\w-\.]{1,}\@([\da-zA-Z-]{1,}\.){1,}[\da-zA-Z-]{2,3}$

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD09
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.97 Email Business 1

This property contains the primary business e-mail address for a contact.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of an e-mail address for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

An example of a regular expression which may partially validate e-mail addresses is: ^[\w-\.]{1,}\@([\da-zA-Z-]{1,}\.){1,}[\da-zA-Z-]{2,3}$

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD0A
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.98 Email Business 2

This property contains the secondary business e-mail address for a contact.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of an e-mail address for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

An example of a regular expression which may partially validate e-mail addresses is: ^[\w-\.]{1,}\@([\da-zA-Z-]{1,}\.){1,}[\da-zA-Z-]{2,3}$

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD0B
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.99 Email Others

This property identifies a list of additional e-mail addresses attributed to a contact.

This property shall contain a string structured as:
EMail 1 description,email 1;EMail 2 description,email 2;
Where "EMail description" is a string describing the address, and "email" is the e-mail address itself (without whitespace). Neither the email description nor the email may contain either the ',' or ';' character, those are strictly used for separation of values within the string. No other restrictions may be placed on the format of either the e-mail description or e-mail value.

An example of a valid value for this property is:

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD0C
Datatype	2	2	UINT16	0x4003 (AINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.100 Phone Number Primary

This property contains the primary phone number for a contact. If multiple phone number properties are supported for a contact, this property may contain the same contents as a more specific property.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of a phone number for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD0D
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.101 Phone Number Personal

This property contains the primary personal phone number for a contact.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of a phone number for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD0E
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.102 Phone Number Personal 2

This property contains the secondary personal phone number for a contact.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of a phone number for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD0F
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.103 Phone Number Business

This property contains the primary business phone number for a contact.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of a phone number for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD10
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.104 Phone Number Business 2

This property contains the secondary business phone number for a contact.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of a phone number for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD11
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.105 Phone Number Mobile

This property contains the primary mobile phone number for a contact.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of a phone number for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD12
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.106 Phone Number Mobile 2

This property contains the secondary mobile phone number for a contact.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of a phone number for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD13
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.107 Fax Number Primary

This property contains the primary fax number for a contact. If multiple fax number properties are supported for a contact, this property may contain the same contents as a more specific property.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of a phone number for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD14
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.108 Fax Number Personal

This property contains the primary personal fax number for a contact.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of a phone number for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD15
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.109 Fax Number Business

This property contains the primary business fax number for a contact.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of a phone number for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD16
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.110 Pager Number

This property contains the primary pager number for a contact.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of a phone number for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD17
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.111 Phone Number Others

This property identifies a list of additional phone numbers attributed to a contact.

This property shall contain a string structured as:
Phone number 1 description,phone number;Phone number 2 description,phone number 2;
Where "Phone number description" is a string describing the type of phone number, and "phone number" is the phone number itself (without whitespace). Neither the description nor the value may contain either the ',' or ';' character, which are strictly used for separation of values within the string. No other restrictions may be placed on the format of either the phone number description or value.

An example of a valid value for this property is:

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD18
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.112 Primary Web Address

This property contains the URL identifying the primary web address associated with a contact.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of a URL for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

An example of a regular expression which may identfy valid URLs is:
((http:)?/{0,2}([^/\?#\[\];:&=\+\$,]*\.)+([^/\?#\[\];:&=\+\$,]{2,3}))(/[^<>])*

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD19
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.113 Personal Web Address

This property contains the URL identifying the personal web address associated with a contact.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of a URL for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

An example of a regular expression which may identfy valid URLs is:
((http:)?/{0,2}([^/\?#\[\];:&=\+\$,]*\.)+([^/\?#\[\];:&=\+\$,]{2,3}))(/[^<>])*

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD1A
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.114 Business Web Address

This property contains the URL identifying the business web address associated with a contact.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of a URL for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

An example of a regular expression which may identfy valid URLs is:
((http:)?/{0,2}([^/\?#\[\];:&=\+\$,]*\.)+([^/\?#\[\];:&=\+\$,]{2,3}))(/[^<>])*

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD1B
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.115 Instant Messenger Address

This property contains the primary instant messenger identifier associated with a contact.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of a URL for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

If multiple instant messenger identifiers are supported on a single contact, the device may use the Regular Expression field of the Object Property Description dataset to associate a particular instant messenger property with an instant messenger host. In this case, the regular expression shall restrict the value of this property to only those values which will operate with that instant messenger host. If the device does this, it is recommended that at least one instant messenger identifier accept arbitrary values.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD1C
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.116 Instant Messenger Address 2

This property contains the secondary instant messenger identifier associated with a contact.

This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of a URL for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

If multiple instant messenger identifiers are supported on a single contact, the device may use the Regular Expression field of the Object Property Description dataset to associate a particular instant messenger property with an instant messenger host. In this case, the regular expression shall restrict the value of this property to only those values which will operate with that instant messenger host. If the device does this, it is recommended that at least one instant messenger identifier accept arbitrary values.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD1D
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.117 Instant Messenger Address 3

This property contains the tertiary instant messenger identifier associated with a contact. This property shall contain a Regular Expression FORM in its Object Property Description dataset. If there is no constraint on the length or structure of a URL for this device, this regular expression may be a null string, indicating that any string is supported. Alternatively, the regular expression may be ".*", indicating that all strings are supported.

If multiple instant messenger identifiers are supported on a single contact, the device may use the Regular Expression field of the Object Property Description dataset to associate a particular instant messenger property with an instant messenger host. In this case, the regular expression shall restrict the value of this property to only those values which will operate with that instant messenger host. If the device does this, it is recommended that at least one instant messenger identifier accept arbitrary values.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD1E
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x05 RegularExpression form

B.2.118 Postal Address Personal Full

This property identifies the full postal address for a contact. This is the address as it is written, and contains no enforced formatting rules. The length of this property may be limited by the MaxLength field in the LongString FORM fields of the Object Property Description dataset describing this property.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD1F
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.119 Postal Address Personal Line 1

This property contains the first line of the personal postal address of this contact. In the United States, this usually includes the street number and street name or post office box number.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD20
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.120 Postal Address Personal Line 2

This property contains the first line of the personal postal address of this contact. In the United States, this usually includes the apartment number, or a further qualification of the address.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD21
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.121 Postal Address Personal City

This property contains the city of the personal postal address for this contact.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD22
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.122 Postal Address Personal Region

This property contains the region of the personal postal address of this contact. In the United States, this would identify the state.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD23
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.123 Postal Address Personal Postal Code

This property contains a personal country-specific postal code for this contact. In the United States, this would be of the form 98052-8300.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD24
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.124 Postal Address Personal Country

This property contains the personal country of the postal address of this contact.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD25
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.125 Postal Address Business Full

This property identifies the full business postal address for a contact. This is the address as it is written, and contains no enforced formatting rules. The length of this property may be limited by the MaxLength field in the LongString FORM fields of the Object Property Description dataset describing this property.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD26
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.126 Postal Address Business Line 1

This property contains the first line of the business postal address of this contact. In the United States, this usually includes the street number and street name or post office box number.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD27
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.127 Postal Address Business Line 2

This property contains the first line of the business postal address of this contact. In the United States, this usually includes the apartment number, or a further qualification of the address.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD28
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.128 Postal Address Business City

This property contains the city of the business postal address for this contact.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD29
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.129 Postal Address Business Region

This property contains the region of the business postal address of this contact. In the United States, this would identify the state.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD2A
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.130 Postal Address Business Postal Code

This property contains a business country-specific postal code for this contact. In the United States, this would be of the form 98052-8300.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD2B
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.131 Postal Address Business Country

This property contains the country of the business postal address of this contact.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD2C
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.132 Postal Address Other Full

This property identifies an extra full postal address for a contact. This is the address as it is written, and contains no enforced formatting rules. The length of this property may be limited by the MaxLength field in the LongString FORM fields of the Object Property Description dataset describing this property.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD2D
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.133 Postal Address Other Line 1

This property contains the first line of an extra postal address of this contact. In the United States, this usually includes the street number and street name or post office box number.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD2E
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.134 Postal Address Other Line 2

This property contains the first line of an extra postal address of this contact. In the United States, this usually includes the apartment number, or a further qualification of the address.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD2F
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.135 Postal Address Other City

This property contains the city of an extra postal address for this contact.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD30
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.136 Postal Address Other Region

This property contains the region of an extra postal address of this contact. In the United States, this would identify the state.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD31
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.137 Postal Address Other Postal Code

This property contains an extra country-specific postal code for this contact. In the United States, this would be of the form 98052-8300.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD32
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.138 Postal Address Other Country

This property contains the country of an extra postal address of this contact.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD33
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.139 Organization Name

This property identifies the primary organization to which a contact belongs. For many contacts, this maty be the name of the company at which they are employed.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD34
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.140 Phonetic Organization Name

This property identifies the phonetic name of the primary organization to which a contact belongs. For many contacts, this maty be the name of the company at which they are employed.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD35
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.141 Role

This property identifies the role, position or job title of a contact. An example is "Software Engineer".

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD36
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.142 Birthdate

This property identifies the birthdate of a contact.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD37
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x03 DateTime form

B.2.143 Message To

This property identifies a list of addresses to which a message has been (or is intended to be) sent. This property shall contain a single string with a semicolon-delimited list of recipients, the type of recipient dependent on the type of message. For example, the Message To property of an e-mail object would contain a semicolon-delimited list of e-mail addresses. The Message To property of an SMS object would contain a semicolon-delimited list of phone numbers.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD40
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.144 Message CC

This property identifies a list of addresses to which a message has been (or is intended to be) copied to. Recipients who are "copied" on a message are those not directly addressed by the message. The specific usage of the "CC" field is dependent on the type of object to which this property is attached.

This property shall contain a single string with a semicolon-delimited list of copied recipients, the type of recipient dependent on the type of message. For example, an e-mail object would contain a semicolon-delimited list of e-mail addresses in this property. An SMS may contain a semicolon-delimited list of phone numbers.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD41
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.145 Message BCC

This property identifies a list of addresses to which a message has been (or is intended to be) blind copied to. Recipients who are "blind copied" on a message are those who receive a message, but who are not identified in the message recieved by any other recipients. The specific usage of the "BCC" field is dependent on the type of object to which this property is attached.

This property shall contain a single string with a semicolon-delimited list of blind copied recipients, the type of recipient dependent on the type of message. For example, an e-mail object would contain a semicolon-delimited list of e-mail addresses in this property. An SMS may contain a semicolon-delimited list of phone numbers.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD42
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.146 Message Read

This property identifies whether a message object has been read.

Valid values of this property are defined as follows:
0x0000 This message has been read/viewed.
0x0001 This message has not been read/viewed.

All other values where Bit 15 is 0 are reserved for MTP
All other values where Bit 15 is 1 are the MTP Vendor Extension range

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD43
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x0000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

B.2.147 Message Received Time

This property identifies the date and time when this message object was received by the intended recipient, relative to the device which received it.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD44
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x03 DateTime form

B.2.148 Message Sender

This property identifies the sender of a message, in such a format that it is recognizable as a valid recipient of a message of the object format type of the object to which this property is attached.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD45
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.149 Activity Begin Time

This property identifies the date and time on which an activity (appointment, meeting, task, etc.) begins, relative to the device clock.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD50
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x03 DateTime form

B.2.150 Activity End Time

This property identifies the date and time on which an activity (appointment, meeting, task, etc.) ends, relative to the device clock.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD51
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x03 DateTime form

B.2.151 Activity Location

This property identifies the location where an activity (appointment, meeting, task, etc.) is planned to occur in a human-readable string.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD52
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.152 Activity Required Attendees

This property contains a list of required attendees for an activity (appointment, meeting, task, etc.). This list must be a semicolon-delimited list of recipients identified in a way appropriate to the object type to which this property is attached.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD54
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.153 Activity Optional Attendees

This property contains a list of optional attendees for an activity (appointment, meeting, task, etc.). This list must be a semicolon-delimited list of recipients identified in a way appropriate to the object type to which this property is attached.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD55
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.154 Activity Resources

This property contains a list of resources required for an activity (appointment, meeting, task, etc.), usually identifying a reserved meeting room or A/V equipment. This list must be a semicolon-delimited list of recipients identified in a way appropriate to the object type to which this property is attached.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD56
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.155 Activity Accepted

This property contains a list of invitees for an activity (appointment, meeting, task, etc.) who have accepted an invitation. This list must be a semicolon-delimited list of recipients identified in a way appropriate to the object type to which this property is attached.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD57
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.156 Activity Tentative

This property contains a list of invitees for an activity (appointment, meeting, task, etc.) who have tentatively accepted an invitation. This list must be a semicolon-delimited list of recipients identified in a way appropriate to the object type to which this property is attached.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD58
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.157 Activity Declined

This property contains a list of invitees for an activity (appointment, meeting, task, etc.) who have declined an invitation. This list must be a semicolon-delimited list of recipients identified in a way appropriate to the object type to which this property is attached.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD59
Datatype	2	2	UINT16	0x4004 (AUINT16)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000 (Empty Array)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0xFF LongString form

B.2.158 Activity Reminder Time

This property identifies the date and time a reminder should be issued to a user to alert them to an upcoming activity (appointment, meeting, task, etc), relative to the device which received it.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD5A
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x03 DateTime form

B.2.159 Activity Owner

This property identifies the owner of an activity (appointment, meeting, task, etc.) in a human-readable string formatted in a manner which is appropriate for the object format type of the object to which this property is attached.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD5B
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.160 Activity Status

This property identifies the current human-readable status of an activity (appointment, task, meeting, etc.).

Possible values include:
"In progress"
"Not yet begun"
"50% complete"
etc.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD5C
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.161 Owner

This property identifies the intellectual property owner of a particular piece of content, mediacast, etc.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD5D
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.162 Editor

This property identifies the human editor of the content.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD5E
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.163 Webmaster

This property identifies the email address of the webmaster of the website or service that provided the content. Examples include the webmaster of a mediacast feed or the webmaster of a music provider.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD5F
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.164 URL Source

This property identifies the URL of the source of the content. Examples include the URL of a mediacast or the URL of a music provider's website.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD60
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.165 URL Destination

This property identifies a URL associated with a piece of content. Examples include the URL of lyrics to a song or biography of an artist.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD61
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.166 Time Bookmark

This property specifies a bookmark (in milliseconds) that corresponds to the last position played or viewed on the given media. During playback, this property will change frequently, and those changes should not result in ObjectPropChanged events unless they are caused by actions that are external to both the current session and the regular playback of the object.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD62
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.167 Object Bookmark

This property identifies the object handle of the current object in a collection of objects. Examples include the current object in a mediacast or playlist. During playback, this property will change frequently, and those changes should not result in ObjectPropChanged events unless they are caused by actions that are external to both the current session and the regular playback of the object.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD63
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.168 Byte Bookmark

This property specifies a bookmark (in bytes) that corresponds to the last position played or viewed on the given media. During playback, this property will change frequently, and those changes should not result in ObjectPropChanged events unless they are caused by actions that are external to both the current session and the regular playback of the object.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD64
Datatype	2	2	UINT16	0x0008 (UINT64)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x0000000000000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.169 Last Build Date

This property specifies a date the last time an object was changed or edited. An example is when a series in a mediacast was changed or edited.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD70
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.170 Time to Live

This property specifies the time, in minutes, until the next content update. An example is the time until a mediacast feed should be updated with new content.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD71
Datatype	2	2	UINT16	0x0008 (UINT64)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x0000000000000000
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

B.2.171 Media GUID

This property identifies a unique number or string that is assigned by the initiator. This number is not associated with the PUID, which is generated by the device.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xDD71
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-defined
DefaultValue	4			0x00 (Null String)
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

Appendix C – Device Properties

C.1 Device Property Summary Table

MTP Name	MTP Datacode
Undefined	0x5000
Battery Level	0x5001
Functional Mode	0x5002
Image Size	0x5003
Compression Setting	0x5004
White Balance	0x5005
RGB Gain	0x5006
F-Number	0x5007
Focal Length	0x5008
Focus Distance	0x5009
Focus Mode	0x500A
Exposure Metering Mode	0x500B
Flash Mode	0x500C
Exposure Time	0x500D
Exposure Program Mode	0x500E
Exposure Index	0x500F
Exposure Bias Compensation	0x5010
DateTime	0x5011
Capture Delay	0x5012
Still Capture Mode	0x5013
Contrast	0x5014
Sharpness	0x5015
Digital Zoom	0x5016
Effect Mode	0x5017
Burst Number	0x5018
Burst Interval	0x5019
Timelapse Number	0x501A
Timelapse Interval	0x501B
Focus Metering Mode	0x501C
Upload URL	0x501D
Artist	0x501E
Copyright Info	0x501F
Synchronization Partner	0xD401
Device Friendly Name	0xD402
Volume	0xD403
SupportedFormatsOrdered	0xD404
DeviceIcon	0xD405
Playback Rate	0xD410
Playback Object	0xD411
Playback Container Index	0xD412
Session Initiator Version Info	0xD406

Perceived Device Type	0xD407

C.2 Device Property Descriptions

C.2.1 Undefined

This is not used

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x5000
Datatype	2	2	UINT16	0x0000 (Undefined)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	Device-Defined

C.2.2 Battery Level

The current battery level of a device is represented by the BatteryLevel property.

The battery level is indicated by an unsigned, read-only integer, and constrained by either an Enumeration or Range of integers. The lowest value in the enumeration or range shall indicate the state of having no battery power remaining, and the largest value shall indicate a full battery. The enumeration or range of other allowed values indicate battery levels at which a DevicePropChanged event shall triggered to indicate to the initiator that that level has been reached, and must therefore not be chosen at too granular a level. A value of 0 may used to indicate that the device has an alternate power source.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x5001
Datatype	2	2	UINT16	0x0002 (UINT8)
Get/Set	3	1	UINT8	0x00 (GET)
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	Device-Defined

C.2.3 Functional Mode

The current functional mode of a device may be retrieved and manipulated using this property.

All devices must default to a standard mode. Non-standard modes generally indicate support for a different level of functionality, either a reduced set (such as when in a sleep state) or an advanced mode (such as when running off an alternative power source). The definition of non-standard modes is dependent on the device. Any change in capability caused by a change in the device's functional mode shall be described in an updated DeviceInfo dataset, and this change shall be communicated using a DeviceInfoChanged event (which shall always be sent when device capabilities change.)

This property is described using an Enumeration and is exposed outside of sessions in the corresponding field in the DeviceInfo dataset.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x5002
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

C.2.4 Image Size

This property indicates and controls the height and width of images which are produced or captured by the device.

The value of this property shall take the form of a Unicode, null-terminated string which is structured as: "WxH", where W represents the width of the image desired, and H represents the height. Both the width and the height are represented by unsigned integers. An example would be a value of "640x480" with a null terminator for the string, which represents a width of 640 and a height of 480 pixels.

The allowed values of this property may be represented by either an Enumeration or a Range form, depending on the capabilities of the device. Devices which can smoothly scale image creation may choose to use a range form. A range form for this property shall have as the minimum of the range a value which is the smallest image it can create, and a maximum value of the largest image it can create, with a step value for each. An example of a range implementation would be a Range form with a minimum value of "1x1" (terminated by a null value), a maximum value of "1024x768" (terminated by a null value) and a step of "1x1" (terminated by a null value) indicating that the image can take any intermediate value.

If the device cannot process all possible image capture sizes in a range, it shall implement this using an enumeration, which shall contain a list of all possible dimensions for captured images.

Changing this property may cause the the Free Space In Objects field of the StorageInfo dataset to be updated. When this occurs, the device is required to issue a StorageInfoChanged event to indicate that this has occurred.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x5003
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	Device-Defined

C.2.5 Compression Setting

Objects captured by a device are generally not stored in their raw form, but are compressed to save limited storage space. The Compression Setting property shall indicate the level of compression in use by the device, and the range of values shall be as close as possible to a linear represention of the perceived quality of the compressed content. Smaller values indicate low quality and high compression, while large values indicate high quality and low compression. This specification does not attempt to assign specific values to this property with any absolute benchmarks, so this value is inherently device and codec specific.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x5004
Datatype	2	2	UINT16	0x0002 (UINT8)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	Device-Defined

C.2.6 White Balance

This property identifies how the device weights the different colour channels.

Valid values include:

0x0000 Undefined

0x0001 Manual
The white balance is set directly by using the RGB Gain property, described in section C.2.7 "RGB Gain", and is static until changed.

0x0002 Automatic
The device attempts to set the white balance using some kind of automatic mechanism.

0x0003 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.

0x0004 Daylight
The device attempts to set the white balance to a value that is appropriate for use in daylight conditions.

0x0005 Florescent
The device attempts to set the white balance to a value that is appropriate for use in conditions with a florescent light source.

0x0006 Tungsten
The device attempts to set the white balance to a value that is appropriate for use in conditions with a tungsten light source.

0x0007 Flash
The device attempts to set the white balance to a value that is appropriate for flash conditions.

All other values with Bit 15 set to zero are reserved for PTP
All other values with Bit 15 set to 1 and Bit 14 set to 0 are open for MTP vendor extensions
All other values with Bit 15 set to 1 and Bit 14 set to 1 are reserved for MTP

Field name	Field order	Size (bytes)	Datatype	Value

PropertyCode	1	2	UINT16	0x5005
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

C.2.7 RGB Gain

This property is a Unicode, null-terminated string which represents the current RGB gain setting of the device. This property is structured as “R:G:B”, where R represents the red gain, G represents the green gain and B represents the blue gain. All the gain values are represented by unsigned integers, up to a maximum of sixteen-bit unsigned integers. An example value of “4:2:3” (terminated by a null value) indicates a gain value of 4 for red, 2 for green and 4 for blue. An example value of “2000:1000:1500” (terminated by a null value) indicates a gain value of 2000 for red, 1000 for green and 1500 for blue. These values are relative to each other, and therefore may take on any integer value less than 2^16.

This property may be constrained by either an Enumeration or a Range form. The minimum value would represent the smallest numerical value (typically "1:1:1", null-terminated). Using values of zero for a particular color channel would mean that color 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 granularity 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".

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x5006
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	Device-Defined

C.2.8 F-Number

This property identifies the aperture setting of the lens.

This property contains the F-number scaled by 100. When the device is set to capture using an automatic exposure mode, the setting of this property may cause other properties (such as Exposure Time and Exposure Index) to change. When that happens, the device must issue a DevicePropChanged event to indicate the change. This property is typically only able to be set when the device's Exposure Program Mode property is set to Manual or Aperture Priority.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x5007
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

C.2.9 Focal Length

This property corresponds to the 35mm equivalent focal length in millimeters multiplied by 100.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x5008
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	Device-Defined

C.2.10 Focus Distance

This property contains an unsigned integer corresponding to the focus distance in millimeters. A value of 0xFFFF indicates a setting greater than 655 meters.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x5009
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	Device-Defined

C.2.11 Focus Mode

This property identifies the current focusing mode in use by the device for image capture. Only the values in the following table are defined by this standard.

Valid values include:
0x0000 Undefined
0x0001 Manual
0x0002 Automatic
0x0003 Automatic Macro (close-up)
All values with Bit 15 set to zero are reserved for PTP
All values with Bit 15 set to 1 and Bit 14 set to 0 are open for MTP vendor extensions
All values with Bit 15 set to 1 and Bit 14 set to 1 are reserved for MTP

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x500A
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

C.2.12 Exposure Metering Mode

This property identifies the current exposure metering mode in use by the device for image capture. Only the values in the following table are defined by this standard.

Valid values include:
0x0000 Undefined
0x0001 Average
0x0002 Center-weighted-average
0x0003 Multi-spot
0x0004 Center-spot
All values with Bit 15 set to zero are reserved for PTP
All values with Bit 15 set to 1 and Bit 14 set to 0 are open for MTP vendor extensions
All values with Bit 15 set to 1 and Bit 14 set to 1 are reserved for MTP

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x500B
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

C.2.13 Flash Mode

This property identifies the current flash mode in use by the device for image capture. Only the values in the following table are defined by this standard.

Valid values include:
0x0000 Undefined
0x0001 Auto flash
0x0002 Flash off
0x0003 Fill flash
0x0004 Red-eye auto
0x0005 Red-eye fill
0x0006 External sync
All values with Bit 15 set to zero are reserved for PTP
All values with Bit 15 set to 1 and Bit 14 set to 0 are open for MTP vendor extensions
All values with Bit 15 set to 1 and Bit 14 set to 1 are reserved for MTP

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x500C
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

C.2.14 Exposure Time

This property identifies the current shutter speed of the device in seconds, scaled by 10,000. If the device is set to an automatic exposure program mode, setting this property through SetDeviceProp may cause other properties to change. When that happens, the device must issue a DevicePropChanged event to indicate the change. This property is typically only able to be set when the device's Exposure Program Mode property is set to Manual or Shutter Priority.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x500D
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	Device-Defined

C.2.15 Exposure Program Mode

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 allowed exposure program mode settings supported by the device.

Valid values include:
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 values with Bit 15 set to zero are reserved for PTP
All values with Bit 15 set to 1 and Bit 14 set to 0 are open for MTP vendor extensions
All values with Bit 15 set to 1 and Bit 14 set to 1 are reserved for MTP

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x500E
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

C.2.16 Exposure Index

This property can be used by an image capture device to emulate film speed settings on a digital camera. The settings of this property 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 automatic ISO setting.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x500F
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	Device-Defined

C.2.17 Exposure Bias Compensation

This property allows the set point of an image capture device's auto exposure control to be identified and set.

This is a signed sixteen-bit integer, and represents a scaling factor.
A value of 0 will not change the factory set auto exposure level. The units of this property represent "stops" scaled by a factor of 1000, which enables fractional stop values. For example, a setting of 2000 indicates two stops of additional exposure (four times more energy to the sensor and a brigher image). A setting of -1000 indicates one stop less exposure (half the energy to the sensor and a darker image). The setting values are expressed in APEX units (Additive system of Photographic Exposure). This property may be constrained by an enumeration or a range.

This property is typically only used when the device has an Exposure Program Mode of Manual.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x5010
Datatype	2	2	UINT16	0x0003 (INT16)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	Device-Defined

C.2.18 DateTime

This property identifies the current date and time settings of the device.

The value of this property follows the 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 to 12), DD is the day of the month (01 to 31), T is a constant character, hh is the hours since midnight (00 to 23), mm is the minutes past the hour (00 to 59), 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 the time is relative to a time zone. Appending neither indicates the time zone is unknown. This property shall not be constrained in the DevicePropDesc form fields, as the ISO 8601 specification already describes the allowed values.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x5011
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

C.2.19 Capture Delay

This value describes the time delay which is to be be inserted between the triggering of the image capture and the actual data capture.

This value is represented by an unsigned integer, which represents the capture delay in milliseconds. This property does not describe the time between multiple frames of a burst capture or capture time of a time-lapse capture, which are described by the Burst Interval and Timelapse Interval properties respectively. 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. When no capture delay is desired, this property shall be set to zero.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x5012
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	Device-Defined

C.2.20 Still Capture Mode

This property identifies the type of still capture which will be performed by an image capture initiation.
Valid values include:
0x0000 Undefined
0x0001 Normal
0x0002 Burst
0x0003 Timelapse
All values with Bit 15 set to zero are reserved for PTP
All values with Bit 15 set to 1 and Bit 14 set to 0 are open for MTP vendor extensions
All values with Bit 15 set to 1 and Bit 14 set to 1 are reserved for MTP

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x5013
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

C.2.21 Contrast

This property identifies the perceived contrast of images captured by the device

This property be constrained by either an enumeration or range, with actual values being relative. The smallest value allowed by the range or enumeration form represents the least contrast, while the largest value represents the most contrast. A value in the middle of the range shall be used to represent the normal (default) contrast.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x5014
Datatype	2	2	UINT16	0x0002 (UINT8)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	Device-Defined

C.2.22 Sharpness

This property identifies the perceived sharpness of images captured by this device.

This property may be constrained by either an enumeration or a range. The minimum value allowed by the range or enumeration form represents the least amount of sharpness, while the largest value represents the most sharpness. A value in the middle of the range shall be used to represent normal (default) sharpness.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x5015
Datatype	2	2	UINT16	0x0002 (UINT8)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	Device-Defined

C.2.23 Digital Zoom

This property identifies the effective digital zoom which will be applied to an image capture device's acquired image, scaled by a factor of 10. When no digital zoom is applied, the value of this property shall be equal to 10. A value of 20 indicates a zoom by a factor of 2 (2X), where only ¼ of the possible scene is captured by the camera. This property may be constrained by either an enumeration or a range, with the lowest value indicating the minimum digital zoom (generally 10) and the largest value indicating the maximum digital zoom which the device can apply.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x5016
Datatype	2	2	UINT16	0x0002 (UINT8)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	Device-Defined

C.2.24 Effect Mode

This property allows the image capture device to specify special image acquisition modes.

Valid values include:
0x0000 Undefined
0x0001 Standard (color)
0x0002 Black & White
0x0003 Sepia
All values with Bit 15 set to zero are reserved for PTP
All values with Bit 15 set to 1 and Bit 14 set to 0 are open for MTP vendor extensions
All values with Bit 15 set to 1 and Bit 14 set to 1 are reserved for MTP

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x5017
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

C.2.25 Burst Number

This property identifies the number of images which the device will capture upon the initiation of a burst capture operation.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x5018
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	Device-Defined

C.2.26 Burst Interval

This property identifies the time delay in milliseconds between subsequent image capture operations in a burst capture operation.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x5019
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	Device-Defined

C.2.27 Timelapse Number

This property indicates the number of images which will be captured when a time-lapse capture is begun.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x501A
Datatype	2	2	UINT16	0x0004 (UINT16)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	Device-Defined

C.2.28 Timelapse Interval

This property indicates the time delay in milliseconds between captures of a time-lapse capture operation.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x501B
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	Device-Defined

C.2.29 Focus Metering Mode

This property identifies the automatic-focus mechanism currently in use by the device.

All allowed values of this property shall be identified in an enumeration form of the DevicePropDesc dataset.

Valid values include:
0x0000 Undefined
0x0001 Center-spot
0x0002 Multi-spot
All values with Bit 15 set to zero are reserved for PTP
All values with Bit 15 set to 1 and Bit 14 set to 0 are open for MTP vendor extensions
All values with Bit 15 set to 1 and Bit 14 set to 1 are reserved for MTP

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x501C
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

C.2.30 Upload URL

This property describes an Internet URL (Universal Resource Locator) which the initiator may use to upload objects after they have been acquired from the device.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x501D
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

C.2.31 Artist

This property contains the name of the owner/operator of this device, and shall be used by the device to populate the "Artist" property in any objects created on this device.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x501E
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

C.2.32 Copyright Info

This property contains the copyright notification which shall be used to populate the "Copyright" property on any objects created by this device.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0x501F
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

C.2.33 Synchronization Partner

This property gives a human-readable description of a synchronization partner for a device. A synchronization partner can be either another device, a software application on a device, or a server over the network. Typically, for a device to PC connection, this is the name of the PC.

This property may also be used by a synchronization process to recognize that it is the partner for this device, so that it may modify its behavior appropriately, but in doing so, should assume that the property will need to be human-readable. This is because this property may be exposed in UI in an operating system.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xD401
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	0x01 (Get/Set)
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

C.2.34 Device Friendly Name

This property gives a human-readable description of the device, for use in an initiating device's user interface.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xD402
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	0x01 (Get/Set)
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

C.2.35 Volume

This identifies (and is used to set) the current volume of the device, and is an unsigned 32-bit integer. The allowed values of this property device shall be identified by a range form defined in the DevicePropDesc dataset defining this property. Values for this property are always based at 0, and a value of 0 indicates that the device is muted.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xD403
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	0x01 (Get/Set)
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x01 Range form

C.2.36 SupportedFormatsOrdered

This property identifies whether a device is indicating its supported production & consumption object formats in order of preference.

Valid values include:
0x00 Unordered
0x01 Ordered
All values with Bit 7 set to 1 are open for MTP vendor extensions
All values with Bit 7 set to 0 are reserved for MTP

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xD404
Datatype	2	2	UINT16	0x0002 (UINT8)
Get/Set	3	1	UINT8	0x00 (GET)
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

C.2.37 DeviceIcon

This property identifies a .ICO icon object that represents the device to an initiator. The specification for a .ICO is located here: http://msdn2.microsoft.com/en-us/library/ms997538.aspx (page title: "Icons in Win32").

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xD405
Datatype	2	2	UINT16	0x4002 (AUINT8)
Get/Set	3	1	UINT8	Device-Defined
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

C.2.38 Playback Rate

This identifies the current speed of playback, identified linearly. It is a signed 32-bit integer, which identifies the speed in thousandths. Thus, a value of 1000 indicates that the playback shall proceed at full speed. A value of 500 indicates that playback shall be at half-speed. A value of -1000 indicates that playback shall be in reverse at full speed. A value of 0 indicates that the device is paused.

A complete list of allowed playback rates for an object shall be contained in an enumeration of allowed values defined in the DevicePropDesc dataset defining this property. This list shall always include the values 1000 and 0.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xD410
Datatype	2	2	UINT16	0x0005 (INT32)
Get/Set	3	1	UINT8	0x01 (Get/Set)
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x02 Enumeration form

C.2.39 Playback Object

This identifies the object currently being played back on the device, identified by Object Handle. This property has two special values. A value of 0x00000000 indicates that the device is currently stopped, and no media file is being consumed.

Devices which support playlist or album objects shall allow this property to contain a reference to an album or playlist. If a device supports these object types, as well as playback control, it must also support the Playback Container Index Device Property. If this property contains an album or playlist object, it indicates that the device is currently playing back the contents of that album or playlist.

Whenever the object being played back is updated on the device (due to the previous object finishing playback, user input on the device, or active control on another active session) the device shall indicate this by initiating a DevicePropChanged event for this property.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xD411
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	0x01 (Get/Set)
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

C.2.40 Playback Container Index

When playing content, the Playback Object device property may contain a container object (album, playlist, etc.) rather than the actual object being consumed. In this case, it is important to expose the specific object in that playback container which is being consumed. The object being played is identified by its index within Object References array of that playback container, and that index is contained in this property. Recall that arrays in MTP are zero-based (so a value of 0x00000000 in this property indicates that the first ObjectHandle in the Object References array is being consumed).
If the Playback Object does not represent a container object, this property shall always contain a value of 0x00000000.

When the playback container index changes, such as during a song change, a DevicePropChanged event shall be sent to notify the initiator.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xD412
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	0x01 (Get/Set)
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

C.2.41 Playback Position

This identifies the current time offset of the object currently being played back in milliseconds. During playback, this property will change frequently, and those changes shall not result in DevicePropChanged events unless they are caused by actions external to both the current session and the regular playback of the object.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xD413
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	0x01 (Get/Set)
DefaultValue	4			Device-Defined
CurrentValue	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

C.2.42 Session Initiator Version Info

This property describes the version information of the session initiator. The value of this property shall take the form of a Unicode, null-terminated string that is formatted as the User Agent string in HTTP 1.1 spec (RFC 2068). The initiator shall set this device property directly after GetDeviceInfo is called. This will give the device a chance to adjust behavior before additional operations occur within the session. GetDeviceInfo is called to determine if this property is supported by the device.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xD406
Datatype	2	2	UINT16	0xFFFF (STRING)
Get/Set	3	1	UINT8	0x01 (Get/Set)
DefaultValue	4			Device-Defined
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

The following example shows a Windows OS version and a class driver version.

Example: "`Windows/6.0.5330.0 MTPClassDriver/6.0.5330.0`**"**

C.2.43 Perceived Device Type

This property allows an Initiator to determine the device type of the Responder, as perceived by the user. This property is intended to be used by the Initiator to graphically represent the Responder; a Device Icon (0xD405) is intended to be preferred over Perceived Device Type. Perceived Device Type may also be used to represent device capabilities or functionality.

Value	Perceived Device Type
0x00000000	Generic
0x00000001	Still Image/Video Camera
0x00000002	Media (Audio/Video) Player
0x00000003	Mobile Handset
0x00000004	Video Player
0x00000005	Personal Information Manager / Personal Digital Assistant
0x00000006	Audio Recorder

All values with Bit 15 set to zero are reserved for MTP.
All values with Bit 15 set to 1 are open for MTP vendor extensions.

Field name	Field order	Size (bytes)	Datatype	Value
PropertyCode	1	2	UINT16	0xD407
Datatype	2	2	UINT16	0x0006 (UINT32)
Get/Set	3	1	UINT8	0x00 (GET)
DefaultValue	4			Device-Defined
GroupCode	5	4	UINT32	Device-defined
FormFlag	6	1	UINT8	0x00 None

Appendix D – Operations

D.1 Operation Summary Table

Operation Name	Operation Code
GetDeviceInfo	0x1001
OpenSession	0x1002
CloseSession	0x1003
GetStorageIDs	0x1004
GetStorageInfo	0x1005
GetNumObjects	0x1006
GetObjectHandles	0x1007
GetObjectInfo	0x1008
GetObject	0x1009
GetThumb	0x100A
DeleteObject	0x100B
SendObjectInfo	0x100C
SendObject	0x100D
InitiateCapture	0x100E
FormatStore	0x100F
ResetDevice	0x1010
SelfTest	0x1011
SetObjectProtection	0x1012
PowerDown	0x1013
GetDevicePropDesc	0x1014
GetDevicePropValue	0x1015
SetDevicePropValue	0x1016
ResetDevicePropValue	0x1017
TerminateOpenCapture	0x1018
MoveObject	0x1019
CopyObject	0x101A
GetPartialObject	0x101B
InitiateOpenCapture	0x101C
GetObjectPropsSupported	0x9801
GetObjectPropDesc	0x9802
GetObjectPropValue	0x9803
SetObjectPropValue	0x9804
GetObjectReferences	0x9810
SetObjectReferences	0x9811
Skip	0x9820

D.2 Operation Descriptions

D.2.1 GetDeviceInfo

This operation returns the DeviceInfo dataset, as defined in section 5.1.1 "DeviceInfo Dataset Description." This dataset provides identifying information about the device, such as model and serial number, as well as describing the capabilities of the device. This operation is commonly the first operation called by an initiator upon connecting to a responder for the first time.

Operation Code	0x1001
Operation Parameter 1	None
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	DeviceInfo dataset
Data Direction	R->I
ResponseCode Options	OK, Parameter_Not_Supported
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

This operation may be called outside of a session. When used outside a session, both the SessionID and TransactionID in the OperationRequest dataset must be 0x00000000.

D.2.2 OpenSession

This operation creates a new session for communication between the Initiator and Responder. Sessions are described in more detail in section 4.4 "Sessions." All operations require a session to exist, unless otherwise specified.

Operation Code	0x1002
Operation Parameter 1	SessionID
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	None
Data Direction	N/A
ResponseCode Options	OK, Parameter_Not_Supported, Invalid_Parameter, Session_Already_Open, Device_Busy
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

If this operation is called and an active session has already been opened, a response of Session_Already_Open shall be returned, and the first response parameter shall contain the SessionID of the open session. This response shall also be returned if the session ID passed in the first parameter is already in use in a different session. If the SessionID passed in the first parameter is equal to 0x00000000, the operation shall fail and shall return a response of Invalid_Parameter.

D.2.3 CloseSession

This operation closes an active session. All stateful information pertaining to the session being closed shall be discarded.

Operation Code	0x1003
Operation Parameter 1	None
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	None
Data Direction	N/A
ResponseCode Options	OK, Session_Not_Open, Invalid_TransactionID, Parameter_Not_Supported
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

D.2.4 GetStorageIDs

This operation returns a list of StorageIDs of storages on this device. StorageIDs are defined in more detail in section 5.2.1 “Storage IDs”.

Operation Code	0x1004
Operation Parameter 1	None
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	StorageID array
Data Direction	R->I
ResponseCode Options	OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Parameter_Not_Supported
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

Removable storages with no inserted media shall be returned in the dataset returned by this operation as well, though they would contain a value of 0x0000 in the lower 16 bits indicating that they are not present.

D.2.5 GetStorageInfo

This operation returns the StorageInfo dataset for the storage identified by the StorageID in the first parameter. The StorageInfo dataset is defined in section 5.2.2, "Storage Info Dataset".

Operation Code	0x1005
Operation Parameter 1	StorageID
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	StorageInfo dataset
Data Direction	R->I
ResponseCode Options	OK, Session_Not_Open, Invalid_TransactionID, Access_Denied, Invalid_StorageID, Store_Not_Available, Parameter_Not_Supported
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

D.2.6 GetNumObjects

This operation returns the number of objects on the device, or the subset defined by the first three parameters.

Operation Code	0x1006
Operation Parameter 1	StorageID
Operation Parameter 2	[ObjectFormatCode]
Operation Parameter 3	[ObjectHandle of Association for which number of children is needed]
Operation Parameter 4	None
Operation Parameter 5	None
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 Parameter 1	NumObjects
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

The first parameter contains the StorageID of the storage for which the number of objects is desired. A value of 0xFFFFFFFF may be used to indicate that an aggregated total across all storages shall be returned. If a storage is specified and the storage is unavailable, this operation shall return Store_Not_Available.

The second parameter is optional, and contains an Object Format datacode. Object Formats are described in section 4, "Object Formats". If the second parameter contains a non-0x00000000 value, it specifies that a count of objects of a certain object format is desired. If the parameter is not used, it shall contain a value of 0x00000000 and objects shall be counted regardless of their object format. If this parameter is not supported, the responder shall return a response code of Specification_By_Format_Unsupported.

The third parameter may be used to restrict the count of objects returned by this operation to objects directly contained in a particular folder (Association). If this parameter contains a non-0x00000000 value, the responder shall return a count of objects which have as their ParentObject the folder (Association) identified by this parameter. If the number of objects contained in the root of a storage is desired, a value of 0xFFFFFFFF may be passed in this operation, indicating that only those objects with no ParentObject (i.e., objects in the root of the storage) shall be returned. If the first parameter indicates that all storages are included in this query, then a value of 0xFFFFFFFF shall return a count of all objects at the root level of any storage. If this parameter is unused, it shall contain a value of 0x00000000.

If the third parameter is unsupported and a non-0x00000000 value is sent in this operation, a response of Parameter_Unsupported shall be returned. If the use of the third parameter is supported, but the value contained does not reference an actual object on the device, a response of Invalid_ObjectHandle shall be returned. If the use of the third parameter is supported and it contains a valid Object Handle, but the object referenced is not of type Association, then a response of Invalid_ParentObject shall be returned.

D.2.7 GetObjectHandles

This operation returns an array of Object Handles referencing the contents of the device.

Operation Code	0x1007
Operation Parameter 1	StorageID
Operation Parameter 2	[ObjectFormatCode]
Operation Parameter 3	[ObjectHandle of Association or hierarchical folder for which a list of children is needed]
Operation Parameter 4	None
Operation Parameter 5	None
Data	ObjectHandle array
Data Direction	R->I
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 Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

The first parameter contains the StorageID of the storage for which the list of Object Handles is desired. A value of 0xFFFFFFFF may be used to indicate that a list of Object Handles of all objects on all storages shall be returned. If a storage is specified and the storage is unavailable, this operation shall return Store_Not_Available.

The second parameter is optional, and contains an Object Format datacode. Object Formats are described in Appendix A – Object Formats. If the second parameter contains a non-0x00000000 value, it specifies that a list of object handles referencing objects of a certain object format is desired. If the parameter is not used, it shall contain a value of 0x00000000 and objects shall be included in the response dataset regardless of their object format. If use of this parameter is not supported, and a non-zero value is passed, the Responder shall return a response code of Specification_By_Format_Unsupported.

The third parameter may be used to restrict the list of objects returned by this operation to objects directly contained in a particular folder (Association). If this parameter contains a non-0x00000000 value, the responder shall return a list of objects which have as their ParentObject the folder (Association) identified by this parameter. If the number of objects contained in the root of a storage is desired, a value of 0xFFFFFFFF may be passed in this operation, indicating that only those objects with no ParentObject are to be returned. If the first parameter indicates that all storages are included in this query, then a value of 0xFFFFFFFF shall return a list of all objects at the root level of all storages. If this parameter is unused, it shall contain a value of 0x00000000.

If the third parameter is unsupported and a non-0x00000000 value is sent in this operation, a response of Parameter_Unsupported shall be returned. If the use of the third parameter is supported, but the value contained does not reference an actual object on the device, a response of Invalid_ObjectHandle shall be returned. If the use of the third parameter is supported and it contains a valid Object Handle, but the object referenced is not of type Association, then a response of Invalid_ParentObject shall be returned.

D.2.8 GetObjectInfo

This operation returns the ObjectInfo dataset for the object identified by the Object Handle in the first parameter. The ObjectInfo dataset is defined in section 5.3.1, "Object Info Dataset".

Operation Code	0x1008
Operation Parameter 1	ObjectHandle
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	ObjectInfo dataset
Data Direction	R->I
ResponseCode Options	OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Invalid_ObjectHandle, Store_Not_Available, Parameter_Not_Supported
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

D.2.9 GetObject

This object retrieves the binary data component of an object from the device. This will generally be preceded by either a GetObjectInfo operation or GetObjectPropList operation(s) to first identify the object's type and descriptive information, but this is not required.

Operation Code	0x1009
Operation Parameter 1	ObjectHandle
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	Object Binary Data
Data Direction	R->I
ResponseCode Options	OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Invalid_ObjectHandle, Invalid_Parameter, Store_Not_Available, Incomplete_Transfer, Access_Denied, Parameter_Not_Supported
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

This object retrives the binary data component of an object from the device. This will generally be preceeded by either a GetObjectInfo operation or GetObjectPropList operation(s) to first identify the object's type and descriptive information, but this is not required.

Objects with no data component (having a binary size 0), such as associations or abstract playlists cannot be retrieved with this operation.

If the object handle in the first parameter refers to a folder (Association) object, the Responder shall respond with an Invalid_ObjectHandle response. If the object handle in the first parameter does not refer to an object on the device, then the responder shall respond with an Invalid_ObjectHandle response. If the first parameter references an object which is not of type Association, but which has a size of 0, then this operation shall succeed, and an object of size 0 shall be returned.

D.2.10 GetThumb

This operation retrieves a thumbnail for an image object on the responder.

The Representative Sample object property may be used as a more flexible alternative for devices which support object properties, but this operation must be supported for PTP-compatibiliy.

Operation Code	0x100A
Operation Parameter 1	ObjectHandle
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	ThumbnailData
Data Direction	R->I
ResponseCode Options	OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Invalid_ObjectHandle, Thumbnail_Not_Present, Invalid_ObjectFormatCode, Store_Not_Available, Parameter_Not_Supported
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

D.2.11 DeleteObject

This operation deletes a data object from the responder.

Operation Code	0x100B
Operation Parameter 1	ObjectHandle
Operation Parameter 2	[ObjectFormatCode]
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	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, Access_Denied
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

The first parameter contains an object handle which references the object to be deleted. Write-protected objects cannot be deleted by this operation. If the first parameter contains a value of 0xFFFFFFFF, then all objects on the device able to be deleted shall be deleted. If a value of 0xFFFFFFFF is passed in the first parameter, and some subset of objects are not deleted (but at least one object is deleted), a response of Partial_Deletion shall be returned. If the object handle in the first parameter does not reference a valid object on the device, the responder shall return an Invalid_ObjectHandle response. If the first parameter identifies an object on the responder, but the object is on a storage which is read-only and does not allow deletion, then the response Store_Read_Only shall be returned.

If the first parameter contains an object handle for a folder object (Association), then all objects in that association shall be deleted. Any folder objects contained in that folder shall also be deleted. If the first parameter contains a folder (Association) object, and that folder contains an object which cannot be deleted, the parent folder of the object which cannot be deleted shall also not be deleted.

The second parameter is optional, and contains an Object Format datacode. Object Formats are described in section 4, “Object Formats”. If the second parameter contains a non-0x00000000 value and the first parameter contains a value of 0xFFFFFFFF, it specifies that all objects on the device of the object format specified in this parameter shall be deleted. If the responder does not support this parameter and it contains a non-0x00000000 value, it shall return a response code of Specification_By_Format_Unsupported.

If all objects identified by the first two parameters are protected and cannot be deleted (as identified by the ProtectionStatus field of the object’s ObjectInfo dataset or the ProtectionStatus object poperty), a response code of Object_WriteProtected shall be returned. If some, but not all, objects are successfully deleted, then a response code of Partial_Deletion must always be returned.

D.2.12 SendObjectInfo

This is the first operation sent when an initiator wishes to send a new object to a responder. When objects are sent to a responder, the ObjectInfo dataset precedes the data component to give the responder context for the transfer, allowing resources to be allocated, and verifying to the initiator that the object should be able to be sent successfully. This operation is usually followed by a SendObject operation, as described in Appendix D.2.13 SendObject". A successful completion of this operation indicates that the responder is ready and able to receive the object described in the ObjectInfo dataset.

Operation Code	0x100C
Operation Parameter 1	[Destination StorageID on responder]
Operation Parameter 2	[Parent ObjectHandle on responder where object shall be placed]
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	ObjectInfo dataset
Data Direction	I->R
ResponseCode Options	OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Access_Denied, Invalid_StorageID, Store_Read_Only, Object_Too_Large, Store_Full, Invalid_ObjectFormatCode, Store_Not_Available, Parameter_Not_Supported, Invalid_ParentObject, Invalid_Dataset, Specification_Of_Destination_Unsupported
Response Parameter 1	Responder StorageID in which the object will be stored
Response Parameter 2	Responder parent ObjectHandle in which the object will be stored
Response Parameter 3	Response Parameter3: responder's reserved ObjectHandle for the incoming object
Response Parameter 4	None
Response Parameter 5	None

The first parameter is optional, and indicates the store on the responder on which the following object shall be stored. If this parameter is included and the responder cannot place the described object in that store, this operation shall fail with a response code of Store_Not_Available, Store_Read_Only or Store_Full. If the responder does not support this parameter, a response code of Specification_Of_Destination_Unsupported shall be returned. If this parameter contains a value of 0x00000000, the responder may choose on which store it will save the object.

The second parameter is optional, and indicates the folder (Association) into which this object shall be placed. If the second parameter contains a non-0x00000000 value, the storage of that parent object must also be specified in the first parameter. If the responder does not support the use of this parameter, it shall fail this operation with a response code of Specification_Of_Destination_Unsupported. If an Initiator receives a response of Specification_Of_Destination_Unsupported when specifying both the first and second parameter, it may try again while specifying only the first parameter if desired. If the object handle included in the second parameter does not reference a valid object on the device, a response of Invalid_ObjectHandle shall be returned. If the object handle included in the second parameter does not reference a folder (Association) object, the responder shall return a value of Invalid_ParentObject.

If the responder cannot accept an object based upon information in the SendObjectInfo dataset (where there is not already an appropriate response), such as an invalid filename, the error code Invalid_Dataset shall be used.

If the initiator wishes to place an object in the root of a given storage, it shall indicate the desired storage in the first parameter and include a value of 0xFFFFFFFF in the second parameter.

Upon the successful completion of this operation, the responder shall be prepared to receive a SendObject operation to receive the binary data for the specified object. If the ObjectInfo dataset sent in the data phase of this operation indicates that the size in bytes of the object to be sent is greater than 0, then the next operation called in the same session is intended to be a SendObject operation. If the next operation sent in the same session is not a SendObject operation, the responder shall not retain the sent or ObjectInfo dataset. If the following SendObject operation does not successfully execute, the ObjectInfo dataset passed in this operation shall be retained until the successful completion of a SendObject operation.

An object handle issued during a successful SendObjectInfo or SendObjectPropList operation should be reserved for the duration of the MTP session, even if there is no successful SendObject operation for that handle.

If the ObjectInfo dataset sent in the data phase of this operation indicates that the object to be sent has a size of 0, then a response of OK indicates that the object has been sent successfully, and it is not required that this operation be followed by a SendObject operation. However, the responder shall not fail if a SendObject operation follows containing an object of size 0.

Object properties that are get-only (0x00 GET) shall accept values during object creation from the SendObjectInfo operation.

D.2.13 SendObject

This operation is used to send the binary content of an object to the device, and follows a successful SendObjectInfo operation (as described in Appendix D.2.12 SendObjectInfo). The data object sent in this operation must correspond to the ObjectInfo dataset description sent in the preceeding SendObjectInfo.

Operation Code	0x100D
Operation Parameter 1	None
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	Object Binary Data
Data Direction	I->R
ResponseCode Options	OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Access_Denied, Invalid_StorageID, Store_Read_Only, Object_Too_Large, Store_Full, Invalid_ObjectFormatCode, Store_Not_Available, Parameter_Not_Supported, Invalid_ParentObject
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

If this operation does not follow a successful SendObjectInfo operation in the same session, a response of No_Valid_ObjectInfo shall be returned. If the target destination does not contain sufficient free space for the object sent in the data phase of this operation, a response of Store_Full shall be returned. If the object sent in the data phase of this operation is larger than the size indicated in the ObjectInfo dataset sent in the SendObjectInfo which precedes this operation, this operation shall fail and a response code of Store_Full shall be returned.

If this operation completes successfully, any stored ObjectInfo dataset shall be discarded. If this operation fails for any reason, the ObjectInfo dataset shall be retained and the responder shall remain ready to receive the object.

If for any reason the data transfer fails during data transfer, the responder shall fail this operation with a response code of Incomplete_Transfer.

D.2.14 InitiateCapture

This operation indicates to the responder that it is to produce a new data object according to its current device properties using an object capture mechanism enabled by the device.

Operation Code	0x100E
Operation Parameter 1	[StorageID]
Operation Parameter 2	[ObjectFormatCode]
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	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, Store_Read-Only
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

The first parameter indicates the storage on which the captured object shall be stored. The first parameter is optional, and if it is not used shall contain a value of 0x00000000. If the first parameter does not contain 0x00000000 and the device cannot place the object on the desired storage, the responder shall fail this operation with the appropriate response code (Store_Not_Available, Invalid_StorageID, Store_Full).
The second parameter contains an ObjectFormatCode which indicates the format of the object to be captured. If the second parameter contains a value of 0x00000000 it indicates that the device shall capture an object in whatever format is the default for the device.

The act of creating a new object in response to receiving this operation is asynchronous, and does not occur immediately. An OK response to this operation indicates that the responder accepts the command and will attempt to create a new object. The completion of the capture shall be indicated by sending a Capture_Complete event. If the capture of new object(s) does not fully complete successfully due to insufficient space on the target storage, a Store_Full event shall be generated and a Capture_Complete event shall not. An ObjectAdded event shall also be generated for each object which is created in the execution of the new capture, and that event shall contain the TransactionID of the InitiateCapture operation which triggered its creation. The device should not send any other events during the capture session (other than ObjectAdded) until the Capture_Complete event has been sent.

A separate operation, InitiateOpenCapture, described in section A.2.28, can be used to support dynamically controlled captures that are terminable by the initiator.

Example Single Object InitiateCapture Sequence:
Initiator -> Responder: InitiateCapture Operation
Responder -> Initiator: InitiateCapture Response
Responder -> Initiator: ObjectAdded Event
Responder -> Initiator: CaptureComplete Event
Initiator -> Responder: GetObjectInfo Operation
Responder -> Initiator: ObjectInfo Dataset/Response

Example Multiple Object InitiateCapture Sequence
Initiator -> Responder: InitiateCapture Operation
Responder -> Initiator: InitiateCapture Response
Responder -> Initiator: ObjectAdded Event(1)
Responder -> Initiator: ObjectAdded Event(2)
. . .
Responder -> Initiator: ObjectAdded Event(n-1)
Responder -> Initiator: ObjectAdded Event(n)
Responder -> Initiator: CaptureComplete Event
Initiator -> Responder: GetObjectInfo Operation(1)
Responder -> Initiator: ObjectInfo Dataset/Response(1)
Initiator -> Responder: GetObjectInfo Operation(2)
Responder -> Initiator: ObjectInfo Dataset/Response(2)
. . .
Initiator -> Responder: GetObjectInfo Operation(n-1)
Responder -> Initiator: ObjectInfo Dataset/Response(n-1)
Initiator -> Responder: GetObjectInfo Operation(n)
Responder -> Initiator: ObjectInfo Dataset/Response(n)

D.2.15 FormatStore

This operation formats the media contained in the storage identified by the StorageID in the first parameter.

Operation Code	0x100F
Operation Parameter 1	StorageID
Operation Parameter 2	[FileSystem Format]
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	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 Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

If the storage in the first parameter cannot be formatted, the responder shall return the appropriate response (Store_Read_Only, Invalid_StorageID, Store_Not_Available). If a specified filesystem format is desired, the second parameter may contain a filesystem type as defined in the StorageInfo dataset described in Section 5.2.2 “Storage Info Dataset”.

If the device is unable to format the store due to concurrency issues, or due to a condition which is known to be temporary, a response of Device_Busy shall be returned.

D.2.16 ResetDevice

This operation signals to the device that it is to return to a default state.

Operation Code	0x1010
Operation Parameter 1	None
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	None
Data Direction	N/A
ResponseCode Options	OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Device_Busy
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

This closes all open sessions. This does not affect the state of the device or its contents, so all device properties and object properties shall remain unchanged following a device reset.

If multiple sessions are open on the Responder when this operation is received, the Responder shall send a DeviceReset event to all open sessions except the one in which the ResetDevice operation was sent. These events must be sent prior to resetting.

D.2.17 SelfTest

This operation directs the device to implement a device-specific self-test.

Operation Code	0x1011
Operation Parameter 1	[SelfTest Type]
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	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 Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

The first parameter is used to indicate the type of self-test that shall be performed, according to the following table.

Value	**Description**
0x0000	Default device-specific self-test
All other values with Bit 15 set to 0	Reserved PTP
All values with Bit 15 set to 1 and Bit 14 set to 0	MTP Vendor Extension range
All values with Bit 15 set to 1 and Bit 14 set to 1	Reserved MTP

D.2.18 SetObjectProtection

This operation sets the write-protection status for the data object referred to in the first parameter to the value indicated in the second parameter.

Operation Code	0x1012
Operation Parameter 1	ObjectHandle
Operation Parameter 2	ProtectionStatus
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	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 Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

For a description of the ProtectionStatus field, refer to the ObjectInfo dataset described in section 5.3.1 "ObjectInfo Dataset Description". If the ProtectionStatus field does not hold a valid value, the ResponseCode shall be Invalid_Parameter.

D.2.19 PowerDown

This operation instructs the device to close all active sessions and power down.

Operation Code	0x1013
Operation Parameter 1	None
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	None
Data Direction	N/A
ResponseCode Options	OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Device_Busy, Parameter_Not_Supported
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

D.2.20 GetDevicePropDesc

This operation returns the DevicePropDesc dataset identified by the DevicePropCode in the first parameter.

Operation Code	0x1014
Operation Parameter 1	DevicePropCode
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	DevicePropDesc dataset
Data Direction	R->I
ResponseCode Options	OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Access_Denied, DeviceProp_Not_Supported, Device_Busy, Parameter_Not_Supported
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

D.2.21 GetDevicePropValue

This operation returns the current value of the device property indicated by the DevicePropCode in the first parameter.

Operation Code	0x1015
Operation Parameter 1	DevicePropCode
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	DeviceProp Value
Data Direction	R->I
ResponseCode Options	OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, DeviceProp_Not_Supported, Device_Busy, Parameter_Not_Supported
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

For more information about device properties refer to section 5.1.2 . The GetDevicePropDesc also returns this value contained in the DevicePropDesc dataset returned by that operation, and when both are supported by a device either can be used.

D.2.22 SetDevicePropValue

This operation sets the value of the device property identified by the DevicePropCode in the first parameter.

Operation Code	0x1016
Operation Parameter 1	DevicePropCode
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	DeviceProp Value
Data Direction	I->R
ResponseCode Options	OK, Session_Not_Open, Invalid_TransactionID, Access_Denied, DeviceProp_Not_Supported, ObjectProp_Not_Supported, Invalid_DeviceProp_Format, Invalid_DeviceProp_Value, Device_Busy, Parameter_Not_Supported
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

The property value must conform to the restrictions placed upon it by the device property description dataset. Device property description datasets are defined in detail in section 5.1.2.1.

If the property cannot be set, the responder shall return a value of Access_Denied. If the value is not in a range allowed by the device and described by the DevicePropDesc dataset, the responder shall respond with Invalid_DeviceProp_Value. If the format or simple type of the new device property value does not correspond with the types specified in the DevicePropDesc dataset for this device property, a response of Invalid_DeviceProp_Format shall be returned.

D.2.23 ResetDevicePropValue

This operation sets the value of the device property identified by the DevicePropCode in the first parameter to the default value.

Operation Code	0x1017
Operation Parameter 1	DevicePropCode
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	None
Data Direction	N/A
ResponseCode Options	OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, DeviceProp_Not_Supported, Device_Busy, Parameter_Not_Supported, Access_Denied
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

The default value for a device property is defined in the DevicePropDesc dataset for that device property. DevicePropDesc datasets are described in section 5.1.2.1.

Attempting to Reset a Get-only device property results in the Access_Denied response.

If the first parameter contains a value of 0xFFFFFFFF, all settable device properties, except DateTime (0x5011), shall be reset to their default value.

D.2.24 TerminateOpenCapture

This operation is used in conjunction with the InitiateOpenCapture operation to capture new objects in an open-ended way.

Operation Code	0x1018
Operation Parameter 1	TransactionID
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	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 Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

The first parameter identifies the TransactionID of the InitiateOpenCapture operation which initiated the capture sequence which the initiator wishes to terminate. If the capture has already completed, this operation shall respond with a Capture_Already_Terminated response code. If the first parameter does not contain a valid TransactionID, or does not refer to a transaction which contained InitiateOpenCapture operation, the responder shall return a Invalid_TransactionID response.

D.2.25 MoveObject

This operation changes the location of an object on the device, either by changing the storage on which it is stored, or changing the location in which it is located, or both.

Operation Code	0x1019
Operation Parameter 1	ObjectHandle
Operation Parameter 2	StorageID of store to move object to
Operation Parameter 3	ObjectHandle of the new ParentObject
Operation Parameter 4	None
Operation Parameter 5	None
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_StorageHandle, Store_Full, Partial_Deletion
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

The first parameter identifies the object which is to be moved. If the first parameter does not refer to an object on the device, the responder shall return an Invalid_ObjectHandle response.

The second parameter is required, and identifies the storage to which the object indicated in the first parameter is to be moved. If the target storage cannot be written to, this operation shall fail with an appropriate response code (Store_Not_Available, Store_Read_Only, Invalid_StorageHandle, Store_Full).

The third parameter is optional, and identifies the location on the file hierarchy of the device to which this object is to be moved. If this parameter is unused, it shall contain a value of 0x00000000, and the object shall be moved to the root of the storage indicated by the second parameter.

If some subset of objects are not moved (but at least one object is moved), the response Partial_Deletion shall be returned.

This object does not change the ObjectHandle of the object which is moved.

D.2.26 CopyObject

This operation causes the device to create a copy of the target object and place that copy in a storage and location indicated by the parameters of this operation.

Operation Code	0x101A
Operation Parameter 1	ObjectHandle
Operation Parameter 2	StorageID that the newly copied object shall be placed into
Operation Parameter 3	ObjectHandle of newly copied object's parent
Operation Parameter 4	None
Operation Parameter 5	None
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 Parameter 1	ObjectHandle of new copy of object
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

The first parameter identifies the object which is to be copied. If the first parameter does not refer to an object on the device, the responder shall return an Invalid_ObjectHandle response.

The second parameter is required, and identifies the storage on which the copy of the object indicated in the first parameter is to be placed. If the target storage cannot be written to, this operation shall fail with an appropriate response code. (Store_Not_Available, Store_Read_Only, Invalid_StorageHandle, Store_Full)

The third parameter is optional, and identifies the location on the file hierarchy of the device to which the copy of the object indicated in the first parameter is to be placed. If this parameter is unused, it shall contain a value of 0x00000000, and the object shall be placed in the root of the storage indicated by the second parameter.
Following the successful completion of this operation, the ObjectHandle of the new object created is returned in the first response parameter.

D.2.27 GetPartialObject

This operation retrieves a partial object from the device, and may be used in place of the GetObject operation.

Operation Code	0x101B
Operation Parameter 1	ObjectHandle
Operation Parameter 2	Offset in bytes
Operation Parameter 3	Maximum number of bytes to obtain
Operation Parameter 4	None
Operation Parameter 5	None
Data	Object Binary Data
Data Direction	R->I
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 Parameter 1	Actual number of bytes sent
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

This operation applies to all data object types on a device. In the context of this operation, the size fields in the ObjectInfo and Size Object Property represent the maximum size, as opposed to the actual size. This operation is not necessary for objects which do not have a binary component, such as folders or hierarchies.

The operation is identical to GetObject, except that the second and third parameters contain the offset in bytes and the number of bytes to obtain starting from the offset. If the entire object is desired, starting from the offset in the second parameter, the third parameter may be set to 0xFFFFFFFF. The first response parameter shall contain the actual number of bytes of the object sent, not including any wrappers or overhead structures.

D.2.28 InitiateOpenCapture

This operation causes the device to initiate the capture of multiple new data objects as specified by the current device properties.

Operation Code	0x101C
Operation Parameter 1	[StorageID]
Operation Parameter 2	[ObjectFormatCode]
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	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, Store_Read-Only
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

The captured objects shall be saved to the store indicated by the StorageID in the first parameter. If the StorageID is 0x00000000, the location where the objects will be saved is determined by the responder. If the store specified is unavailable or the first parameter contains 0x00000000 and there are no stores available, this operation shall return Store_Not_Available.

Capturing new data objects is an asynchronous operation. If the ObjectFormatCode in the second operation parameter is unspecified (and contains a value of 0x00000000), then the device shall capture an object in a format chosen by the device.

A successful response to the InitiateOpenCapture operation means that the responder has begun to capture one or more objects. When the initiator wishes to terminate the capturing of new objects, it shall send a TerminateOpenCapture operation. The CaptureComplete event shall not be sent at the end of this capture period if it is terminated by the initiator. As new objects are created on the responder, the responder is required to send an ObjectAdded event to the initiator for each object. The ObjectAdded event shall contain the TransactionID of the InitiateOpenCapture operation which initiated the capture of the device.

If the store becomes full while completing this operation, the device shall send a Store_Full event containing the TransactionID of the InitiateOpenCapture operation in progress. In the case of multiple objects being captured, each object shall be treated separately, so any object captured before the store becomes full shall be retained.

Whether an object that was partially captured can be retained and used is a function of the device's behavior and object format. For example, 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. A Store_Full event completes the capture.

Single Object InitiateOpenCapture Sequence
Initiator -> Responder: InitiateOpenCapture Operation
Responder -> Initiator: InitiateOpenCapture Response
Initiator -> Responder: TerminateOpenCapture Operation
Responder -> Initiator: TerminateOpenCapture Response
Responder -> Initiator: ObjectAdded Event
Initiator -> Responder: GetObjectInfo Operation
Responder -> Initiator: ObjectInfo Dataset/Response

Multiple Object InitiateOpenCapture Sequence
Initiator -> Responder: InitiateOpenCapture Operation
Responder -> Initiator: InitiateOpenCapture Response
Responder -> Initiator: ObjectAdded Event(1)*
Responder -> Initiator: ObjectAdded Event(2)
Responder -> Initiator: ObjectAdded Event(n-1)
Responder -> Initiator: ObjectAdded Event(n)
Initiator -> Responder: TerminateOpenCapture Operation
Responder -> Initiator: TerminateOpenCapture Response
Initiator -> Responder: GetObjectInfo Operation(1)
Responder -> Initiator: ObjectInfo Dataset/Response(1)
Initiator -> Responder: GetObjectInfo Operation(2)
Responder -> Initiator: ObjectInfo Dataset/Response(2)
Initiator -> Responder: GetObjectInfo Operation(n-1)
Responder -> Initiator: ObjectInfo Dataset/Response(n-1)
Initiator -> Responder: GetObjectInfo Operation(n)
Responder -> Initiator: ObjectInfo Dataset/Response(n)

D.2.29 GetObjectPropsSupported

This operation returns an ObjectPropCode array of supported object properties for the object format indicated in the first parameter.

Operation Code	0x9801
Operation Parameter 1	ObjectFormatCode
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	ObjectPropCode Array
Data Direction	R->I
ResponseCode Options	OK, Operation_Not_Supported, Device_Busy, Invalid_TransactionID, Invalid_ObjectFormatCode
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

D.2.30 GetObjectPropDesc

This operation returns the appropriate property describing dataset indicated in the first parameter as defined for the object format indicated in the second parameter.

Operation Code	0x9802
Operation Parameter 1	ObjectPropCode
Operation Parameter 2	Object Format Code
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	ObjectPropDesc dataset
Data Direction	R->I
ResponseCode Options	OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Access_Denied, Invalid_ObjectPropCode, Invalid_ObjectFormatCode, Device_Busy
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

D.2.31 GetObjectPropValue

This operation returns the current value of an object property.

Operation Code	0x9803
Operation Parameter 1	ObjectHandle
Operation Parameter 2	ObjectPropCode
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	ObjectProp Value
Data Direction	R->I
ResponseCode Options	OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Invalid_ObjectPropCode, Device_Busy, Invalid_Object_Handle
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

The first parameter is required and identifies the object for which the property is requested.

The second parameter is required and identifies the property that is requested for the object identified in the first parameter.

The size and format of the data returned from this operation shall be determined from the corresponding ObjectPropDesc dataset returned from the GetObjectPropDesc operation.

D.2.32 SetObjectPropValue

This operation sets the current value of the object property.

Operation Code	0x9804
Operation Parameter 1	ObjectHandle
Operation Parameter 2	ObjectPropCode
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	ObjectProp Value
Data Direction	I->R
ResponseCode Options	OK, Session_Not_Open, Invalid_TransactionID, Access_Denied, Invalid_ObjectPropCode, Invalid_ObjectHandle, Device_Busy, Invalid_ObjectProp_Format, Invalid_ObjectProp_Value
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

This operation sets the current value of the object property indicated by parameter 2 for the object indicated by parameter 1 to the value indicated in the data phase of the operation. The format of the property value object sent in the data phase can be determined by the DatatypeCode field of the property's ObjectPropDesc dataset. If the property is not settable, the response Access_Denied shall be returned. If the value is not allowed by the device, Invalid_ObjectProp_Value shall be returned. If the format or size of the property value is incorrect, Invalid_ObjectProp_Format shall be returned.

D.2.33 GetObjectReferences

This operation returns an array of currently valid ObjectHandles.

Operation Code	0x9810
Operation Parameter 1	ObjectHandle
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	ObjectHandle array
Data Direction	R->I
ResponseCode Options	OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Invalid_ObjectHandle, Store_Not_Available
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

If the object handle passed in the first parameter does not refer to a valid object, a response code of Invalid_ObjectHandle shall be returned.

D.2.34 SetObjectReferences

This operation replaces the object references on an object.

Operation Code	0x9811
Operation Parameter 1	ObjectHandle
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	ObjectHandle array
Data Direction	I->R
ResponseCode Options	OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Access_Denied, Invalid_StorageID, Store_Read_Only, Store_Full, Store_Not_Available, Invalid_ObjectHandle, Invalid_ObjectReference
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

This operation replaces the object references on a device with the array of object handles passed in the data phase. The object handles passed in the data phase must be maintained indefinitely, and returned as valid object handles referencing the same object in later sessions. If any of the object handles in the array passed in the data phase are invalid, the responder shall fail the operation by returning a response code of Invalid_ObjectReference.

If the object handle passed in the first parameter does not refer to a valid object a response code of Invalid_ObjectHandle shall be returned.

If SetObjectReferences is called on an association object with an Association Type of 1 and an Association Description of 1, the device shall fail the operation with the response Access_Denied.

D.2.35 Skip

This operation updates the current object being played back.

Operation Code	0x9820
Operation Parameter 1	Skip Index
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	None
Data Direction	N/A
ResponseCode Options	OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Access_Denied, Store_Not_Available, Invalid_Parameter
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

This operation updates the current object being played back by skipping either ahead or behind in a device-specific playback queue. This operation requires one parameter, containing a signed INT32 value, which indicates the depth and direction into the playback queue to which the current playback object should skip.

A value of 1 indicates that the device shall skip ahead one media object to the object immediately following the object currently identified in the Playback Object device property. A value of -1 indicates that the previous object in the device playback queue shall be loaded as the current playback object. If a device supports this operation, it must support values of [-1,1]. If a value outside of this range is passed in this parameter, and the device is incapable of interpreting it, a response code of Invalid_Parameter shall be returned. If a value of 0 is passed in this parameter, the responder shall fail this operation with a response code of Invalid_Parameter.

Appendix E – Enhanced Operations

E.1 Enhanced Operation Summary Table

Operation Name	Operation Datacode
GetObjectPropList	0x9805
SetObjectPropList	0x9806
GetInterdependentPropDesc	0x9807
SendObjectPropList	0x9808

E.2 Enhanced Operation Descriptions

E.2.1 GetObjectPropList

This operation returns a dataset containing all object properties specified by the query defined by the five parameters.

The primary purpose of this operation is to provide optimized access to object properties without needing to individually query each {object,property} pair. However, it also provides a more flexible querying mechanism for object properties in general, and supercedes the simpler, object property retrieval methods.The primary purpose of this operation is to provide optimized access to object properties without needing to individually query each {object,property} pair. However, it also provides a more flexible querying mechanism for object properties in general, and supercedes the simpler, object property retrieval methods.

Operation Code	0x9805
Operation Parameter 1	ObjectHandle
Operation Parameter 2	[ObjectFormatCode]
Operation Parameter 3	ObjectPropCode
Operation Parameter 4	[ObjectPropGroupCode]
Operation Parameter 5	[Depth]
Data	ObjectPropList dataset
Data Direction	R->I
ResponseCode Options	OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, ObjectProp_Not_Supported, Invalid_ObjectHandle, Group_Not_Supported, Device_Busy, Parameter_Not_Supported, Specification_By_Format_Unsupported, Specification_By_Group_Unsupported, Specification_By_Depth_Unsupported, Invalid_Code_Format, Invalid_ObjectPropCode, Invalid_StorageID, Store_Not_Available
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

The first parameter is required, and defines the object for which properties are requested. A value of 0xFFFFFFFF indicates that all objects are requested. A value of 0x00000000 indicates that all objects at the root level are desired, and may be further specified by the second and/or fifth parameters.

The second parameter is optional, and may be used to request only properties for objects which possess a format specified by the ObjectFormatCode. A value of 0x00000000 indicates that this parameter is not being used, and properties of all Object Formats are desired. If the value is not 0x00000000 and the device does not support specification by ObjectFormatCode, it shall fail the operation by returning a response code with the value of Specificiation_By_Format_Unsupported. For the second parameter, the value 0xFFFFFFFF is reserved for future use.

The third parameter is required and identifies the ObjectPropCode of the property that is being requested. A value of 0xFFFFFFFF indicates that all properties are requested except those with a group code of 0xFFFFFFFF; properties with this group code are defined as being potentially very slow, and shall be retrieved separately.

A value of 0x00000000 in the third parameter indicates that the fourth parameter shall be used. If the value is 0x00000000 and the Responder does not support specification by ObjectPropGroupCode, it shall fail the operation by returning a response code with the value of Specification_By_Group_Unsupported. If both the third and the fourth parameters contain the value 0x00000000, then Parameter_Not_Supported shall be returned.

The final parameter is optional, and allows properties to be queried for all objects at a certain level (or levels) of a folder hierarchy on the device. In this case, properties (as defined by the third and/or fourth parameters) shall be returned for all objects, down to a depth from the top object, including the top object, which is identified in the first parameter. If the first parameter contains a value of 0x00000000, then the Responder shall return property values for objects starting from the root (having no parent object) to the desired depth.

If the value of the fifth parameter is 0x00000000, it indicates that properties for objects are desired to a depth of 0, which returns only the head object (as indicated by the first parameter). If the fifth parameter contains a value of 0x00000000, and the ObjectHandle in the first parameter also contains a value of 0x00000000, the Responder shall return an empty set.

If the final parameter contains a value of 0xFFFFFFFF, the Responder shall return all values for all objects that are contained within the folder hierarchy rooted at the object identified by the first parameter. It should be noted that a value of 0x00000000 in the first two parameters, followed by a value of 0xFFFFFFFF in the fifth parameter, is equivalent to having a value of 0xFFFFFFFF in the first parameter.

If the fifth parameter contains a non-zero value and the property or properties indicated by the third and fourth parameters are not supported for all objects in the desired hierarcy, then the Responder shall return an ObjectPropList dataset containing only the properties which are both requested and supported for each (but not necessarily every) object in the desired hierarchy. If a property is identified by the third parameter which not implemented for any object format type on the device, then a response of ObjectProp_Not_Supported shall be returned.

It is recommended that Responders support specification by depth to at least one level to support file-browsing scenarios. If the Responder does not support specification by depth, or the Responder does not support specification to the desired depth, it shall fail the operation by returning a response code with the value of Specification_By_Depth_Unsupported.

E.2.1.1 ObjectPropList Dataset Table:

Field name	Field order	Size (bytes)	Datatype	Description
NumberOfElements	1	4	UINT32	Count of property quadruples in this dataset.
Element1ObjectHandle	2	4	ObjectHandle	ObjectHandle of the object to which Property1 applies.
Element1PropertyCode	3	2	Datacode	Datacode identifying the ObjectPropDesc describing Property1.
Element1Datatype	4	2	Datacode	This field identifies the DatatypeCode of Property1.
Element1Value	5	DTS	DTS	Value of Property1.
Element2ObjectHandle	6	4	ObjectHandle	ObjectHandle of the object to which Property2 applies.
Element2PropertyCode	7	2	Datacode	Datacode identifying the ObjectPropDesc

				describing Property2.
Element2Datatype	8	2	Datacode	This field identifies the DatatypeCode of Property2.
Element2Value	9	DTS	DTS	Value of Property2.
...				
ElementNObjectHandle	4*N-2	4	ObjectHandle	ObjectHandle of the object to which Property*N* applies.

E.2.2 SetObjectPropList

This operation sets ObjectProperty values contained in the dataset provided.

Operation Code	0x9806
Operation Parameter 1	None
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	ObjectPropList dataset
Data Direction	I->R
ResponseCode Options	OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Access_denied, ObjectProp_Not_Supported, Invalid_ObjectProp_Format, Invalid_ObjectProp_Value, Invalid_ObjectHandle, Device_Busy, ObjectProp_Not_Supported, Store_Not_Available, Store_Full
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

If this operation fails, and all property values sent in the ObjectPropList are not applied successfully, the operation must return a ResponseCode identifying the reason for failing to update the property, and a response parameter indicating the (0-based) index of the property which failed to be applied. The responder must not process any of the object property values that follow the property that failed to update. If none of the properties were able to be set, the response parameter shall contain a value of 0x00000000.

If the operation succeeds, the first response parameter shall contain 0x00000000.

E.2.3 GetInterdependentPropDesc

An Initiator can query for interdependent properties using the GetInterdependentPropDesc operation.

Operation Code	0x9807
Operation Parameter 1	ObjectFormatCode
Operation Parameter 2	None
Operation Parameter 3	None
Operation Parameter 4	None
Operation Parameter 5	None
Data	InterdependentPropDesc dataset
Data Direction	R->I
ResponseCode Options	OK, Operation_Not_Supported, Session_Not_Open, Invalid_TransactionID, Device_Busy, Invalid_Code_Format
Response Parameter 1	None
Response Parameter 2	None
Response Parameter 3	None
Response Parameter 4	None
Response Parameter 5	None

An Initiator can query for interdependent properties using the GetInterdependentPropDesc operation, which returns an array of ObjectPropertyDesc arrays, each describing an allowed collection of ranges. Each array of ObjectPropertyDesc datasets returned gives one possible definition for the interdependent properties contained in that array; properties not found in that array are constrained only by the usual ObjectPropDesc datasets.

The first parameter is required, and defines the object format for which interdependent property codes are desired. A value of 0xFFFFFFFF or 0x00000000 shall not be used.

The operation returns the dataset shown in the following table.

E.2.3.1 InterDependentPropList Dataset Table

Field name	Field order	Size (bytes)	Datatype	Description
NumberOfInterdependencies	1	2	UINT16	Count of arrays of interdependencies to follow
NumberOfPropDescs 1	2	2	UINT16	Count of object property

				description datasets in this interdependency array
ObjectPropDesc Dataset 1,1	3	DTS	See ObjectPropDesc dataset definition	An ObjectPropDesc dataset defining allowed values in this interdependent set
…				
ObjectPropDesc Dataset 1, *n*	*n*+3	DTS	See ObjectPropDesc Dataset Definition	An ObjectPropDesc dataset defining allowed values in this interdependent set
NumberOfPropDescs 2	*n*+4	2	UINT16	Count of object property description datasets in this interdependency array
ObjectPropDesc Dataset 2,1	*n*+5	DTS	See ObjectPropDesc dataset definition	An ObjectPropDesc dataset defining allowed values in this interdependent set.
…				
ObjectPropDesc Dataset 2, *m*	*m*+*n*+4	DTS	See ObjectPropDesc Dataset Definition	An ObjectPropDesc dataset defining allowed values in this interdependent set
…				

This dataset begins with an entry which counts the number of interdependencies which follow.

Each interdependency which follows consists of a set of concatenated ObjectPropDesc datasets whose constraints (as identified in their FORM fields) apply as a group. This set is preceded by a count of the number of ObjectPropDesc datasets which follow in that interdependent set of object properties.

E.2.4 SendObjectPropList

Operation Code	0x9808
Operation Parameter 1	[Destination StorageID on responder]
Operation Parameter 2	[Parent ObjectHandle on responder where object shall be placed]
Operation Parameter 3	ObjectFormatCode
Operation Parameter 4	ObjectSize [most significant 4 bytes]
Operation Parameter 5	ObjectSize [least significant 4 bytes]
Data	ObjectPropList dataset
Data Direction	I->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, Invalid_Dataset, ObjectProp_Not_Supported, Invalid_ObjectProp_Format, Invalid_ObjectProp_Value, Invalid_ObjectHandle, Object_Too_Large, Store_Full, Specification_Of_Destination_Unsupported
Response Parameter 1	Responder StorageID in which the object will be stored
Response Parameter 2	Responder parent ObjectHandle in which the object will be stored
Response Parameter 3	Responder's reserved ObjectHandle for the incoming object
Response Parameter 4	[Index of failed property]
Response Parameter 5	None

This is used as an alternative first operation when the initiator wants to send an object to the responder. This operation sends a modified ObjectPropList dataset from the initiator to the responder.

This operation is sent prior to the SendObject operation, in order to inform the responder about the properties of the object that it intends to send later, and to ask whether the object can be sent to the responder. A response of "OK" implies 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 into which the object shall be stored. If this parameter is specified, and the responder will not be able to store the object in the indicated store, the operation shall fail, and the appropriate response, such as Specification_Of_Destination_Unsupported, Store_Not_Available, Store_Read_Only, or Store_Full shall be used. If this parameter is unused, it shall be set to 0x00000000, and the responder shall decide in which store to place the object, whether that is a responder-determined default location, or the location with the most free space (or possibly the only location with enough free space).

The second parameter is optionally used to indicate where on the indicated store the object is to be placed (the association/folder that the object is to become a child of). If this parameter is used, the first parameter must also be used. If the receiver is unable to place the object as a child of the indicated second parameter, the operation shall fail.

The third parameter identifies the ObjectFormat datacode of the object, which may be required to validate the properties sent in the data phase. If the ObjectFormat code indicated in the third parameter is not supported by the responder (as indicated in the DeviceInfo dataset), the responder shall fail and return an Invalid_ObjectHandle response.

The fourth and fifth parameters together indicate the size of the data object to be sent. If the indicated size is larger than the available space on the device, it shall fail this operation and return a response of Store_Full.

If the problem with the attempted specification is the general inability of the receiving device to allow the specification of destination, the response Specification_of_Destination_Unsupported shall be sent. This response implies 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 shall be used, depending on whether the ObjectHandle did not refer to a valid object or the indicated object is a valid object, but is not an association.

If the root directory of the indicated store is needed, the second parameter shall be set to 0xFFFFFFFF. If this parameter is unused, it shall 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 which store to place the object in, and where to place it within that store.

An object handle issued during a successful SendObjectInfo or SendObjectPropList operation should be reserved for the duration of the MTP session, even if there is no successful SendObject operation for that handle.

Object properties that are get-only (0x00 GET) shall accept values during object creation via the SendObjectPropList command.
If the responder agrees that the object may be sent, it is required to retain this ObjectPropList dataset until the next SendObject, SendObjectPropList or SendObjectInfo operation is performed within the session. If the SendObjectPropList operation succeeds, and the next occurring SendObject operation does not return a successful response, the sent ObjectPropList dataset shall be retained by the responder in case the initiator wants to re-attempt the SendObject operation for that previously successful SendObjectPropList operation. If the initiator wants to resend the ObjectPropList dataset before attempting to resend the object, it may do so. Successful completion of the SendObjectPropList operation conveys that the responder possesses a copy of all sent properties, 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 ObjectPropList dataset, and the object shall not be sent.

For a particular session, the receiving device shall only retain one ObjectPropList or ObjectInfo dataset that is the result of a SendObjectInfo or SendObjectPropList operation in memory at a time. If another SendObjectInfo or SendObjectPropList operation occurs before a SendObject operation, the new ObjectInfo or ObjectPropList shall replace the previously held one. If this occurs, any storage or memory space reserved for the object described in the overwritten ObjectInfo or ObjectPropList dataset shall be freed before overwriting and allocating the resources for the new data. Upon the successful execution of this operation, the next operation called in the same session shall be a SendObject operation. If the next operation sent in the same session is not a SendObject operation, the responder shall not retain the sent ObjectPropList or ObjectInfo dataset.
The first response parameter of this operation shall be set to the StorageID that the responder will store the object into if it is sent. The second response parameter of this operation shall be set to the parent ObjectHandle of the association that the object becomes a child of. If the object is stored in the root of the store, this parameter shall be set to 0xFFFFFFFF.

If the initiator wants to retain an association hierarchy on the responder for the objects it is sending, then the objects must be sent top down, starting with the highest level of the hierarchy, and 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 SendObjectPropList response as the second operation parameter for a child's SendObjectPropList operation.

The dataset sent in this operation is similar to the ObjectPropValueList dataset sent and received in the SetObjectPropList and GetObjectPropList operations respectively, but has additional restrictions on the values of the contained fields. All ObjectHandle fields must contain the value 0x00000000, and all properties defined in this operation will be applied to the object, which is sent in a subsequent SendObject operation. If any properties are inconsistent, that is, the property is either not supported or the value is inconsistent for the sent ObjectFormat, then this operation shall fail with the appropriate response code, and indicate the (0-based) index of the first failed property in the fourth return parameter. If the object size indicated in the data phase of this operation indicates that the object to be sent has a size of 0, then a response of OK indicates that the object has been sent successfully, and it is not required that this operation be followed by a SendObject operation. However, the responder shall not fail the request if a SendObject operation follows containing an object of size 0.

Properties which are contained in the operation parameters (StorageID, ParentObject, ObjectFormat, ObjectSize) shall not be included in the sent dataset.

Appendix F – Responses

F.1 Response Summary Table

Name	Datacode
Undefined	0x2000
OK	0x2001
General_Error	0x2002
Session_Not_Open	0x2003
Invalid_TransactionID	0x2004
Operation_Not_Supported	0x2005
Parameter_Not_Supported	0x2006
Incomplete_Transfer	0x2007
Invalid_StorageID	0x2008
Invalid_ObjectHandle	0x2009
DeviceProp_Not_Supported	0x200A
Invalid_ObjectFormatCode	0x200B
Store_Full	0x200C
Object_WriteProtected	0x200D
Store_Read-Only	0x200E
Access_Denied	0x200F
No_Thumbnail_Present	0x2010
SelfTest_Failed	0x2011
Partial_Deletion	0x2012
Store_Not_Available	0x2013
Specification_By_Format_Unsupported	0x2014
No_Valid_ObjectInfo	0x2015
Invalid_Code_Format	0x2016
Unknown_Vendor_Code	0x2017
Capture_Already_Terminated	0x2018
Device_Busy	0x2019
Invalid_ParentObject	0x201A
Invalid_DeviceProp_Format	0x201B
Invalid_DeviceProp_Value	0x201C
Invalid_Parameter	0x201D
Session_Already_Open	0x201E
Transaction_Cancelled	0x201F
Specification_of_Destination_Unsupported	0x2020
Invalid_ObjectPropCode	0xA801
Invalid_ObjectProp_Format	0xA802
Invalid_ObjectProp_Value	0xA803
Invalid_ObjectReference	0xA804
Group_Not_Supported	0xA805
Invalid_Dataset	0xA806
Specification_By_Group_Unsupported	0xA807
Specification_By_Depth_Unsupported	0xA808

Object_Too_Large	0xA809
ObjectProp_Not_Supported	0xA80A

F.2 Response Descriptions

F.2.1 Undefined
Response Code: 0x2000

This response code is not used.

F.2.2 OK
Response Code: 0x2001

Operation has completed successfully.

F.2.3 General_Error
Response Code: 0x2002

This operation did not complete, and the reason for the failure is not known.

F.2.4 Session_Not_Open
Response Code: 0x2003

Indicates that the session handle identified by the operation dataset for this operation is not a currently open session.

F.2.5 Invalid_TransactionID
Response Code: 0x2004

Indicates that the TransactionID of this operation does not identify a valid transaction.

F.2.6 Operation_Not_Supported
Response Code: 0x2005

This response indicates that an Operation has been called with what appears to be a valid code, but the responder does not support the operation identified by that code. The initiator should only invoke operations contained in the responder's DeviceInfo dataset, so this response should not normally be returned.

F.2.7 Parameter_Not_Supported
Response Code: 0x2006

Indicates that a parameter of an operation contains a non-zero value, but is not supported. This response is different from Invalid_Parameter.

F.2.8 Incomplete_Transfer
Response Code: 0x2007

This response shall be sent when a transfer did not complete successfully, and indicates that data transferred is to be discarded. This response shall not be sent if the transfer was cancelled by the Initiator.

F.2.9 Invalid_StorageID
Response Code: 0x2008

Indicates that one or more StorageIDs sent as parameters of an operation do not refer to actual StorageIDs on the device.

F.2.10 Invalid_ObjectHandle
Response Code: 0x2009

Indicates that one or more ObjectHandles sent as parameters of an operation do not refer to actual Objects on the device. The list of valid ObjectHandles should be requested again, along with any appropriate ObjectInfo datasets.

F.2.11 DeviceProp_Not_Supported
Response Code: 0x200A

Indicates that a DevicePropCode sent as a parameter of an operation appears to be a valid code, but is not supported by the device. The initiator should only attempt to work with Device Properties identified in the DevicePropertiesSupported field of the DeviceInfo Dataset, so this response should not normally be returned.

F.2.12 Invalid_ObjectFormatCode
Response Code: 0x200B

Indicates that the device does not support an ObjectFormatCode supplied in the given context.

F.2.13 Store_Full
Response Code: 0x200C

Indicates that a store identified in this operation is full, and this is preventing the successful completion of that operation.

F.2.14 Object_WriteProtected
Response Code: 0x200D

Indicates that an object referred to by the operation is write-protected.

F.2.15 Store_Read-Only
Response Code: 0x200E

Indicates that a store referred to by the operation is read-only.

F.2.16 Access_Denied
Response Code: 0x200F

This response shall be sent when access to data required by the operation is denied. This shall not be used when the device is busy, but to indicate that if the current state of the device does not change access will continue to be denied.

F.2.17 No_Thumbnail_Present
Response Code: 0x2010

Indicates that a data object exists with the specified ObjectHandle, but a thumbnail cannot be provided for that object.

F.2.18 SelfTest_Failed
Response Code: 0x2011

This shall be sent when the device fails a device-specific self test.

F.2.19 Partial_Deletion
Response Code: 0x2012

Indicates that only a subset of the objects indicated for deletion were actually deleted. This could be caused by some of those objects being write-protected or on read-only stores.

F.2.20 Store_Not_Available
Response Code: 0x2013

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.

F.2.21 Specification_By_Format_Unsupported
Response Code: 0x2014

This response shall be sent when an operation attempts to specify an action only on objects which have a particular format code, but the responder does not support that capability. The operation should be attempted again without specifying by format. When this response is sent, it shall indicate that any future attempts to call the same operation specifying by format will also result in this response.

F.2.22 No_Valid_ObjectInfo
Response Code: 0x2015

This shall be sent when a SendObject operation has been called without the initiator having previously sent a corresponding SendObjectInfo successfully. The initiator must successfully complete a SendObjectInfo operation before attempting another SendObject operation.

F.2.23 Invalid_Code_Format
Response Code: 0x2016

Indicates that a datacode used in this operation does not have the correct format, and is therefore known to be invalid. This response shall be used when the most-significant bits of a datacode does not have the format required for that type of code, and not when the data appears to have the correct type but is invalid for other reasons.

F.2.24 Unknown_Vendor_Code
Response Code: 0x2017

Indicates that the indicated data code has the correct format, but is in a vendor extension range not recognized by the device. This response will typically not occur, because the Initiator can identify the supported vendor extensions by examination of the DeviceInfo dataset.

F.2.25 Capture_Already_Terminated
Response Code: 0x2018

This shall be sent when an operation attempts to terminate a capture session, but that the capture session has already terminated. This response is only used for the TerminateOpenCapture operation, which is only used to terminate open-ended captures.

F.2.26 Device_Busy
Response Code: 0x2019

This response shall be sent when the device is not currently able to process a request because it, or the specified store, is busy. This response implies that the operation may be successful at a later time, but is not possible right now. This response shall not be used to indicate that a store is physically unavailable.

F.2.27 Invalid_ParentObject
Response Code: 0x201A

This response shall be sent when an indicated object is not of type Association, but is required to be of type Association in the context in which it is used, 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, but only for ObjectHandles which refer to actual objects which are not of type Association.

F.2.28 Invalid_DeviceProp_Format
Response Code: 0x201B

This response shall be sent when an attempt is made to set a DeviceProperty, but the DevicePropDesc dataset sent is not the correct size or format.

F.2.29 Invalid_DeviceProp_Value
Response Code: 0x201C

This response shall be sent when an attempt is made to set a DeviceProperty to a particular value, but that value is not allowed by the device.

F.2.30 Invalid_Parameter
Response Code: 0x201D

This response indicates that a parameter of the operation is not a valid value. This response is different from Parameter_Not_Supported, which indicates that no value was expected in this parameter.

F.2.31 Session_Already_Open
Response Code: 0x201E

This response may be sent in resonse to an OpenSession operation. If multiple sessions are supported by the device, this response indicates that a session with the specified SessionID is already open. If multiple sessions are not supported by the device, this response indicates that a session is open and must be closed before another session can be opened.

F.2.32 Transaction_Cancelled
Response Code: 0x201F

This response indicates that the operation was interrupted due to manual cancellation by the initiator.

F.2.33 Specification_of_Destination_Unsupported
Response Code: 0x2020

This response may be sent as a response to a SendObjectInfo operation to indicate that the responder does not support the specification of destination. This response implies that any future attempts to specify the object destination will also fail with the same response.

F.2.34 Invalid_ObjectPropCode
Response Code: 0xA801

Indicates that the device does not support the sent Object Property Code in this context.

F.2.35 Invalid_ObjectProp_Format
Response Code: 0xA802

Indicates that an object property sent to the device is in an unsupported size or type.

F.2.36 Invalid_ObjectProp_Value
Response Code: 0xA803

Indicates that an object property sent to the device is the correct type, but contains a value which is not supported. The supported values shall be identified by the ObjectPropDesc dataset.

F.2.37 Invalid_ObjectReference
Response Code: 0xA804

Indicates that a sent Object Reference is invalid. Either the reference contains an object handle not present on the device, or the reference attempting to be set is unsupported in context.

F.2.38 Invalid_Dataset
Response Code: 0xA806

Indicates that the dataset sent in the data phase of this operation is invalid. While the PTP specification (refer to "USB Still Image Capture Device Definition – July 2000" and the specifications referred to by that document) refers to "Invalid_Dataset", at no time does it specify a response code. Thus the definition of Invalid_Dataset in this MTP specification does not conflict with the PTP specification.

F.2.39 Specification_By_Group_Unsupported
Response Code: 0xA807

May be used as the response to indicate that the responder does not support the specification of groups by the initiator. This response implies that the initiator should not attempt to specify the group code in any future operations, as they will also fail with the same response.

F.2.40 Specification_By_Depth_Unsupported
Response Code: 0xA808

May be used as the response to indicate that the responder does not support the specification of depth by the initiator. This response implies that the initiator should not attempt to specify depth in any future call of the operation which resulted in this response, as they will also fail with the same response.

F.2.41 Object_Too_Large
Response Code: 0xA809

Indicates that the object desired to be sent cannot be stored in the filesystem of the device. This should not be used when there is insufficient space on the storage. For example, a FAT32 system can only support a 4GB object. A 6GB object would receive Object_Too_Large.

F.2.42 ObjectProp_Not_Supported

Response Code: 0xA80A

Indicates that an ObjectPropCode sent as a parameter of an operation appears to be a valid code, but is not supported by the device. The initiator should only attempt to work with Object Properties identified as supported by the responder, so this response should not normally be returned.

F.2.43 Group_Not_Supported

Response Code: 0xA805

Indicates that an Object Property group code sent as a parameter of an operation appears to be a valid code, but is not supported by the device. The initiator should only attempt to work with Object Property group codes identified as supported by the responder, so this response should not normally be returned.

Appendix G – Events

G.1 Event Summary Table

MTP Name	Event Datacode
Undefined	0x4000
CancelTransaction	0x4001
ObjectAdded	0x4002
ObjectRemoved	0x4003
StoreAdded	0x4004
StoreRemoved	0x4005
DevicePropChanged	0x4006
ObjectInfoChanged	0x4007
DeviceInfoChanged	0x4008
RequestObjectTransfer	0x4009
StoreFull	0x400A
DeviceReset	0x400B
StorageInfoChanged	0x400C
CaptureComplete	0x400D
UnreportedStatus	0x400E
ObjectPropChanged	0xC801
ObjectPropDescChanged	0xC802
ObjectReferencesChanged	0xC803

G.2 Event Descriptions

G.2.1 Undefined

Event Code: 0x4000
Parameter 1: None
Parameter 2: None
Parameter 3: None

This event code is undefined, and is not used.

G.2.2 CancelTransaction

Event Code: 0x4001
Parameter 1: None
Parameter 2: None
Parameter 3: None

This event is used to initiate the cancellation of a transaction.It is strongly recommended to utilize USB cancelation functionality in preference to this protocol level cancelation. When an Initiator or Responder receives this event, it shall cancel the transaction identified by the TransactionID in the event dataset. If the transaction has already completed, this event shall be ignored.

After receiving a CancelTransaction event from the initiator during an object transfer, the responder shall send an Transaction_Cancelled response for the transfer which was in progress.

G.2.3 ObjectAdded

Event Code: 0x4002
Parameter 1: ObjectHandle
Parameter 2: None
Parameter 3: None

This event indicates that a new data object has been added to the device. The first parameter of this event shall contain the ObjectHandle assigned by the device to the new object. If more than one object has been added, each new object shall generate its own ObjectAdded event. This event shall not be issued by the appearance of a new store on the device, which shall instead cause the generation of a StoreAdded event.

G.2.4 ObjectRemoved

Event Code: 0x4003
Parameter 1: ObjectHandle
Parameter 2: None
Parameter 3: None

This event indicates that a data object has been removed from the device for reasons external to the current session. The handle of the removed object shall be contained in the first parameter of this event. If more than one object is removed, each removed object shall generate its own ObjectRemoved event. If the data object was removed because of a previous operation issued in the current session, no event shall be isued. This event shall not be issued by the removal of a store from the device, which shall instead cause the generation of one StoreRemoved event with the appropriate PhysicalStorageID.

G.2.5 StoreAdded

Event Code: 0x4004
Parameter 1: StorageID
Parameter 2: None
Parameter 3: None

This event indicates that a new store has been added to the device. If this is a new physical store which contains only one logical store, then the complete StorageID of the new store shall be contained in the first parameter. If the new store contains more than one logical store, then the first parameter shall be set to 0x00000000 and the initiator should retrieve a new list of StorageIDs using the GetStorageIDs operation. Any new StorageIDs discovered should result in the appropriate invocations of GetStorageInfo operations.

G.2.6 StoreRemoved

Event Code: 0x4005
Parameter 1: StorageID
Parameter 2: None
Parameter 3: None

The indicated stores are no longer available. The opposing device may assume that the StorageInfo datasets 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 that has been removed is only a single logical store within a physical store, the entire StorageID shall 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 (for example, removal of an ejectable media device that contains multiple partitions), the first parameter shall contain the PhysicalStorageID in the most significant sixteen bits, with the least significant sixteen bits set to 0xFFFF.

G.2.7 DevicePropChanged

Event Code: 0x4006
Parameter 1: DevicePropCode
Parameter 2: None
Parameter 3: None

A property changed on the device due to something external to this session. The appropriate property dataset should be requested from the opposing Responder.

G.2.8 ObjectInfoChanged

Event Code: 0x4007
Parameter 1: ObjectHandle
Parameter 2: None
Parameter 3: None

This event indicates that the ObjectInfo dataset for a particular object has changed, and that it should be requested again.

G.2.9 DeviceInfoChanged

Event Code: 0x4008
Parameter 1: None
Parameter 2: None
Parameter 3: None

This event indicates that the capabilities of the Responder have changed, and that the DeviceInfo should be requested again. This may be caused by the Responder going into or out of a sleep state, or by the Responder losing or gaining some functionality.

G.2.10 RequestObjectTransfer

Event Code: 0x4009
Parameter 1: ObjectHandle
Parameter 2: None
Parameter 3: None

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 that intrinsically use pull mode.

G.2.11 StoreFull

Event Code: 0x400A
Parameter 1: StorageID
Parameter 2: None
Parameter 3: None

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

G.2.12 DeviceReset

Event Code: 0x400B
Parameter 1: None
Parameter 2: None
Parameter 3: None

This event only needs to be supported for devices that support multiple sessions or if 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.

G.2.13 StorageInfoChanged

Event Code: 0x400C
Parameter 1: StorageID
Parameter 2: None
Parameter 3: None

This event is used when information in the StorageInfo dataset 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.

G.2.14 CaptureComplete

Event Code: 0x400D
Parameter 1: TransactionID
Parameter 2: None
Parameter 3: None

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 the result of this asynchronous operation. This operation is not used for InitiateOpenCapture operations.

G.2.15 UnreportedStatus

Event Code: 0x400E
Parameter 1: None
Parameter 2: None
Parameter 3: None

When an initiator receives this event, it is responsible for doing whatever is necessary to ensure that its knowledge of the responder is current. This may include re-obtaining information such as individual datasets or ObjectHandle lists, or may even result in the session being closed and re-opened.

G.2.16 ObjectPropChanged

Event Code: 0xC801
Parameter 1: ObjectHandle
Parameter 2: ObjectPropCode
Parameter 3: None

This event is used to indicate that an object property value on the Responder has changed, without that change being performed by the initiator. The parameters passed indicate which property on which object has been updated.

G.2.17 ObjectPropDescChanged

Event Code: 0xC802
Parameter 1: ObjectPropCode
Parameter 2: ObjectFormatCode
Parameter 3: None

This event indicates that an object property description dataset has been updated, indicating some change on the device. The parameters passed with this event identify the ObjectPropDesc dataset which has changed.

G.2.18 ObjectReferencesChanged

Event Code: 0xC803
Parameter 1: ObjectHandle
Parameter 2: None
Parameter 3: None

This event is used to indicate that the references on an object have been updated. The object handle in the first parameter identifies the object whose references have changed. When objects are deleted by the Initiator and the Responder cleans up any references to the now-deleted object, the Responder does not need to send this event for the resulting reference updates.

Appendix H – USB Optimizations

Sending >4GB Binary Objects

As outlined in section 4.3 MTP Transactions, transactions in MTP take place in three phases: Operation, Data and Response. The USB implementation defined by ISO 15740 requires that the data communicated in a given phase be contained in a container structure, outlined in Appendix D of the ISO 15740. This generic container contains an entire transfer phase; multiple containers cannot be combined. In PTP, any short packet indicates the end of a particular phase (a NULL packet if ContainerLength divides the USB Packet Size.)

USB Generic Container Dataset

Byte Offset	Length (Bytes)	Field Name	Description
0	4	ContainerLength	Total amount of data to be sent (including this header)
4	2	ContainerType	Defines the type of this container: 0x0000 Undefined 0x0001 Command 0x0002 Data 0x0003 Response 0x0004 Event
6	2	Code	Operation, Response or Event Code as defined in the MTP specification.
8	4	TransactionID	See section 4.3.3 Transaction IDs.
12	ContainerLength-12	Payload	The data which is to be sent in this phase.

This container structure restricts the total size of the data transmitted in a phase to a size able to be defined by a 4-byte field (approx 4GB). In order to send a larger data object during a data phase, a value of **0xFFFFFFFF** shall be contained in the ContainerLength field. This may only be performed during a data phase; the restriction that the Command, Response and Event phases cannot contain more than (2^{32}-1) bytes remains.

Sending a >4GB Object with a SendObject Operation
In the likely scenario where the generic container represents the data phase of a SendObject operation, the following additional restrictions apply:

It is the responsibility of the initiator to ensure that there is sufficient space on the responder to contain the sent data object on the target storage; failure to do so will result in the data transfer failing when the incoming object has overflowed the allocated storage on the device.

It is the responsibility of the responder to first ensure that it has space for at least a 4GB data object. If not, it shall respond appropriately. If the initiator specified a target storage for the object being sent, the responder shall attempt to place it on that storage, and fail if there is insufficient space on that storage with the appropriate response code. Then, assuming that a file of at least 4Gb may be sent, it shall accept the object.

Retrieving a >4GB Object with a GetObject Operation
The other likely scenario for large data transfer is retrieving an object from a responder.

When retrieving an object, the size of the object to be retrieved shall be identified using the ObjectCompressedSize Object Property, which is not limited to 32 bits. That size shall be used to identify the size of the incoming data transfer, not including the size of the generic container header. (Add 12 bytes to the value in the ObjectCompressedSize object property to determine the value that would be placed in the ContainerLength field if it were not limited to 32 bits.)

Splitting the Header and Data during the Data Phase
As outlined in section 4.3 MTP Transactions, transactions in MTP take place in three phases: Operation, Data and Response. The USB implementation defined by ISO 15740 requires that the data communicated in a given phase be contained in a container structure, outlined in Appendix D of the ISO 15740, and also reproduced in the previous section. This generic container contains an entire transfer phase; multiple containers cannot be combined. In PTP, any short packet indicates the end of a particular phase (a NULL packet if ContainerLength divides the USB Packet Size).

This artificial header presents a difficulty for devices that wish to write an incoming datastream directly to the device, or that wish to pipe data directly to an outgoing datastream.

An MTP responder may overcome this by separating the header from the payload and sending/receiving it in a short packet preceding the payload. Devices that choose to do this must **always** manage these packets consistently. That is, all data phases (all USB data transfers where the ContainerType = 0x0002) must have a single packet containing 12

bytes, which has only the header which is followed by the payload beginning with a new packet. This applies both to data sent to the device and retrieved from the device.

Operations, Responses and Events remain unchanged, and shall **never** have their generic container header split from the payload.

If an MTP responder implementation chooses to take advantage of this option, it does not need to indicate this directly in any way. Rather, it is the responsibility of the MTP initiator to determine whether the MTP device is separating the header from the payload, based upon observed behavior. The suggested method is to use a known required operation (such as GetDeviceInfo) which has a data phase containing known data, and based upon the format of the structure returned in the data phase, determine how to handle future data phases.

If the data payload transferred in a data phase of an operation is empty, that is, there is no data but a data phase is defined, then the data phase consists of a single generic container header (which identifies the total amount of data to be 12 bytes: the size of the generic container header), and it shall not be followed by a Zero-Length Packet packet (unless the container itself is a multiple of wMaxPacketSize for the endpoint).