Standard of the Camera & Imaging Products Association

CIPA DC-X005- 2005

PTP - IP (tentative name)

Short Abstract: This document describes a proposed transport of ISO-15740 "Picture Transfer Protocol" over TCP/IP networks.

28.July. 2005

Published by

Camera & Imaging Products Association

Disclaimer and Copyright Notice

Table of Contents

1 Introduction

1.1 Purpose

This document describes a proposed implementation of ISO-15740/PIMA 15740:2000/Picture Transfer Protocol (“PTP”) [1] over IP networks.

PTP is designed to provide standard communication with digital still photography devices. The protocol specifies standard image referencing behaviour, operations, responses, events, device properties, datasets, and data formats to ensure interoperability and also provides optional operations and formats, as well as extension mechanisms.

1.2 Formatting Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in "Key words for use in RFCs to Indicate Requirement Levels" [5].

1.3 References

The following table lists the references used in this document.

ID/Description
[1] PTP protocol, PIMA 15740:2000
[2] Computer Networks, Andrew S. Tanenbaum, ISBN: 0130661023, Prentice Hall Aug 09, 2002
[3] UPnP Standard, http://www.upnp.org/resources/documents.asp
[4] ZeroConf Standard, http://www.zeroconf.org
[5] Formatting Conventions, http://www.faqs.org/rfcs/rfc2119.html

1.4 Acronyms and abbreviations

Abbreviations and acronyms are explained in the following table.

Abbreviation	Meaning
TCP	Transfer Control Protocol
UDP	User Datagram Protocol
IP	Internet Protocol
PnP	Plug and Play
PTP	Picture Transfer Protocol
PTP-IP	IP Picture Transfer Protocol
PIMA	Photographic and Imaging Manufacture Association
DHCP	Dynamic Host Configuration Protocol
v4LL	IPv4 Link-Local Addressing

1.5 Contributors

Petronel Bigioi – FotoNation Ireland
George Susanu – FotoNation Ireland
Alexei Pososin – FotoNation Ireland
Denis McHugh – FotoNation Ireland
Eran Steinberg – FotoNation US
Yury Prilusky – FotoNation US

1.6 Picture Transfer Protocol ("PTP") Terms

PTP specification defines device roles, operations, image formats and other camera specific data types and structures. In the TCP/IP implementation of PTP, all terms and definitions are used and understood as defined in PTP.

1.6.1 Device roles

From a communication point of view a device can act as an Initiator or a Responder. The Initiator is the device that initiates operation requests to the Responder device. The Responder is a device that responds to those operation requests by sending requested data and/or responses related to the operation. A device may have the capability to be only an Initiator, only a Responder or both, but not on the same device connection.

The operation requests can only be initiated by the Initiator device. The events are used by the Responder device to notify the Initiator about a change in its state.

1.6.2 Sessions

According to PTP protocol, a session is defined as a logical connection between two imaging devices over which operation and event transactions can be communicated. According to PTP specification, a Responder can support multiple concurrent sessions, each of the session having its own state.

A session is opened when the Initiator device requests the OpenSession operation transaction and the request is successfully completed with a valid response from the Responder device.

A session is closed when the Initiator device requests the CloseSession operation transaction and the request is successfully completed with a valid response from the Responder device. A session is also closed when the connection between the Initiator and Responder device is lost (e.g. communication timeouts, etc…).

Most of the operations require an open session context to be executed in. However, sessions do not need to be open to obtain device capabilities via the GetDeviceInfo operation.

Devices that can support multiple sessions MUST be able to keep them asynchronous and opaque to each other.

1.6.3 Session ID

Usually, the Session ID is intended for the Responder to be able to dispatch in proper order requests coming from different Initiators, if the same transport level communication channels are used. For transports where different communication channels are open, for each session, the Session ID is not required, since the Responder will be able to deal with multiple sessions.

PTP-IP transport implementation of PTP will not carry the SessionID over to the Responder. The responder MAY limit the number of concurrent PTP sessions by controlling the maximum number of PTP-IP connections it accepts.

If the Initiator's PTP-IP layer finds out that a new PTP-IP connection is not accepted by the Responder it issues a transport-specific error code to the PTP Layer. After the transport specific connection has been established, the Initiator MAY issue GetDeviceInfo and OpenSession to the remote Responder. At this stage, the Responder MAY return a Device_Busy PTP error as response to an OpenSession (if the Responder has limitations on the number of concurrent PTP sessions that it can support). In this case, the Initiator SHOULD release the PTP connection(s) and re-try at a later stage.

1.6.4 Transactions

All operations defined in PTP happen via transactions. Transactions are atomic operations. The operation transaction requests are issued in the direction from the Initiator to the Responder device and only one operation transaction at a time can take place within a session.

Generally, transactions consist of three phases, as shown on Figure 1. Depending on the operation transaction, the data phase may not be present. If the data phase is present, data may be sent either from the Initiator to the Responder (I -> R), or from the Responder to the Initiator (R -> I), but never in both directions for the same operation. The operation request and response phases are always present.

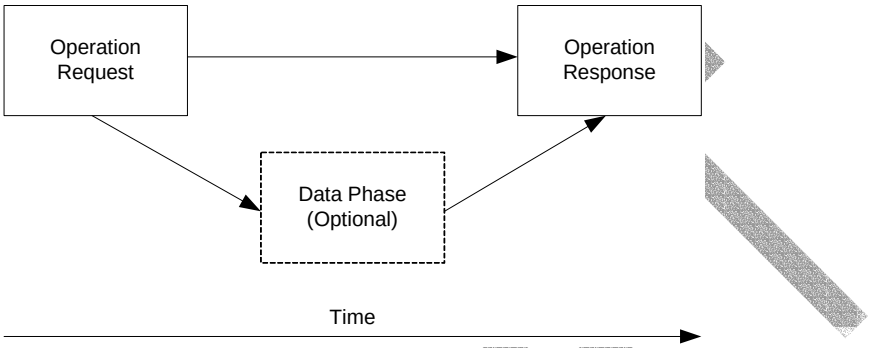


Figure 1: Transaction State Diagram

For each session there can be only one operation transaction in progress, i.e. operation transactions are considered synchronous within a session unless otherwise indicated.

The asynchronous type of operation transactions (e.g. InitiateCapture), are handled by making the operation initiation synchronous. The response to this type of operation transactions indicates only the success or failure of the operation initiation, and asynchronous events are used to handle the communication of new objects becoming available on the device at a later point in time.

If an operation transaction request can't be accommodated because of another previously invoked asynchronous operation still being executed, the device SHALL indicate that it is busy, using the Device_Busy response code. The Initiator device is required to re-issue the operation at a later time.

1.6.5 TransactionID

The TransactionID is a number used to identify transactions on a particular session, as defined in the PTP Clause 9.3.1 [1]. The TransactionID numbers are unique only within a given session and are represented as a continuous sequence of numbers in numerical order starting from 0x00000001. The TransactionID = 0x00000000 and ID = 0xFFFFFFFF are reserved values and are not considered valid TransactionID within an open session.

1.7 Conformance with PTP

In the PTP Clause 7 [1] these requirements are defines for the underlying transport layer.

- PTP Clause 7.1 – Disconnection Event. This clause of PTP requires that the transport layer has to report device disconnection to higher layers. This is achieved by implementation of the PTP-IP layer, by generating a DeviceDisconnection notification.

- PTP Clause 7.2 – Reliable, Error Free Channel. This clause of PTP requires a reliable, error free channel. In the PTP-IP, this is achieved by using TCP connections configured with Keep Alive option
- PTP Clause 7.3 – Asynchronous Event Support. This clause of PTP requires support for Asynchronous Events. In the PTP-IP, the Event Connection is defined to deliver asynchronous events and is asynchronous to the data/communication channel.
- PTP Clause 7.4 – Device Discovery and Enumeration. This clause of PTP requires support a device discovery and enumeration service. In the PTP-IP, the device discovery and enumeration is achieved via general-purpose solutions available for IP networks. Alternatively, a custom UDP based device discovery protocol [5] can be used.
- PTP Clause 7.5 – Specific Transport. This clause of PTP requires an image device transport implementation conformance with the particular used transport specification issued by the associate authority. This document specifies how the PTP-IP transport is used on top of TCP/IP stack.

2 PTP implementation over TCP/IP

2.1 IP Picture Transfer Protocol (PTP-IP) Model

In PTP-IP implementation, the communication between the User application and PTP-IP layer is based on PTP transactions model, as defined in PTP Clause 9.3 [1]. This communication model is shown in Figure 2

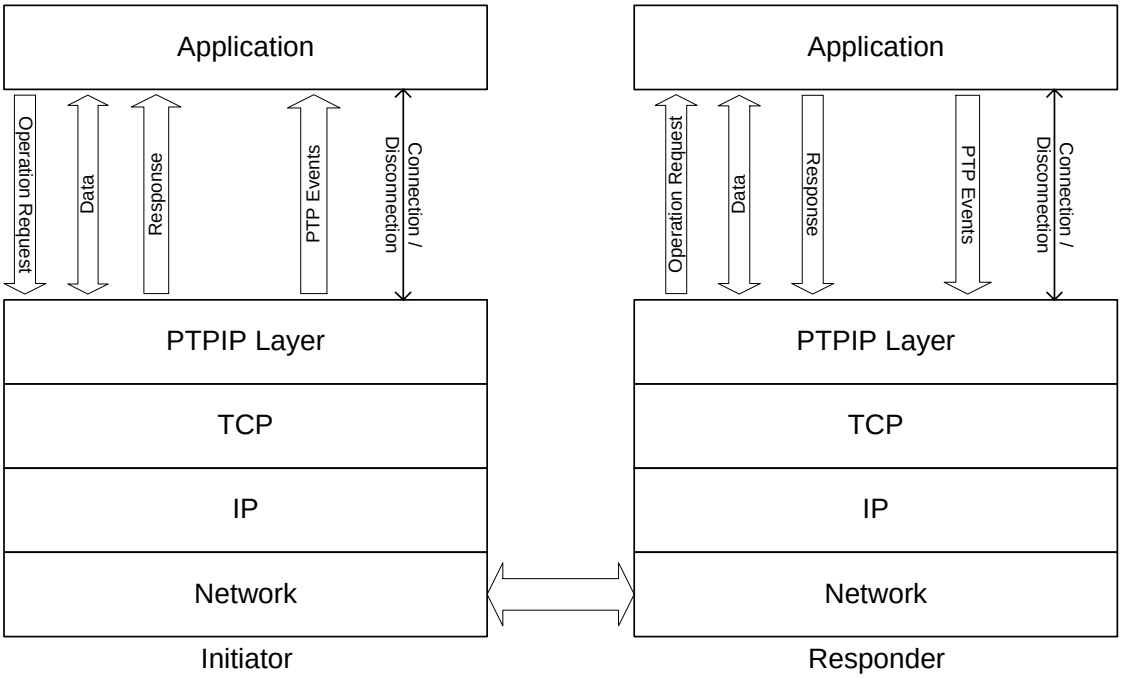


Figure 2: PTP-IP Communication Model

The purpose of this document is to define the PTP-IP communication protocol. It is not in the scope of this document to specify the exact implementation of the interaction between the application and PTP-IP layer (the programmatic interface, the API). This document only specifies the communication between two PTP-IP entities in the way that the interoperability can be achieved. However, the interface between the PTP Layer (user application) and the PTP-IP layer MAY consist of the following primitives:

- Operation Request is initiated by the application in the Initiator device whenever an action is requested. Operation Requests are accepted by the application in the Responder device. An Operation Request typically contains a set of parameters related to the operation.
- Operation Responses is initiated by the application in the Responder device as a reaction to the previously accepted Operation Request. Also an Operation Response is expected for each Operation Request initiated by the application in the Initiator device. An Operation Response always contains the requested operation result code and may contain a set of parameters.
- Data-In is data being transferred from the Responder to the Initiator.
- Data-Out is data being transferred from the Initiator to the Responder.
- Events are notifications about changes in the state of the Responder. Events are initiated by the Responder.

- Device Connect / Disconnect are platform dependent type of events. Those events are not communicated between the Initiator and Responder but are generated by each local PTP-IP layer when the device connection or disconnection is detected.

The types of Operation and Event transactions are defined in the PTP Clause 10,11 and 12 [1]. The PTP Clause 14 [1] determines a set of Operation and Events for device standard conformance.

2.2 Transport Model

PTP Clause 7 [1] defines the PTP standard transport requirements. The PTP-IP implementation is based on the use of TCP (Transfer Control Protocol) layer as its transport layer, as shown in Figure 3

Like PTP, also PTP-IP expects from its transport layer reliable, error free communication channels. TCP (in the TCP/IP protocol stack) is the natural transport layer to satisfy those requirements. TCP is a stream-based transport layer that provides multiple communication channels (TCP connections) and error-free data delivery.

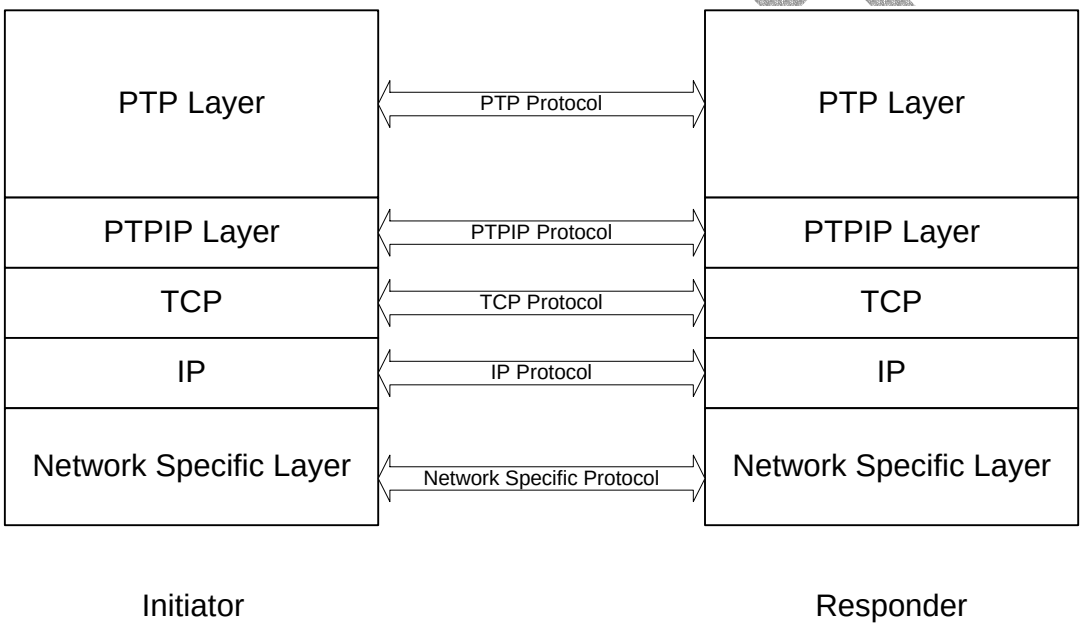


Figure 3: PTP-IP place in the PTP Communication stack

In PTP-IP, all communications between two devices happen via two TCP connections (logical data channels), as shown in Figure 4.

The first connection is used for Operation Request, Response and Data transaction packets. It is called Command/Data Connection. The second TCP connection is used for the Event transaction packets. It is called Event Connection. Both connections are used for Cancel packets.

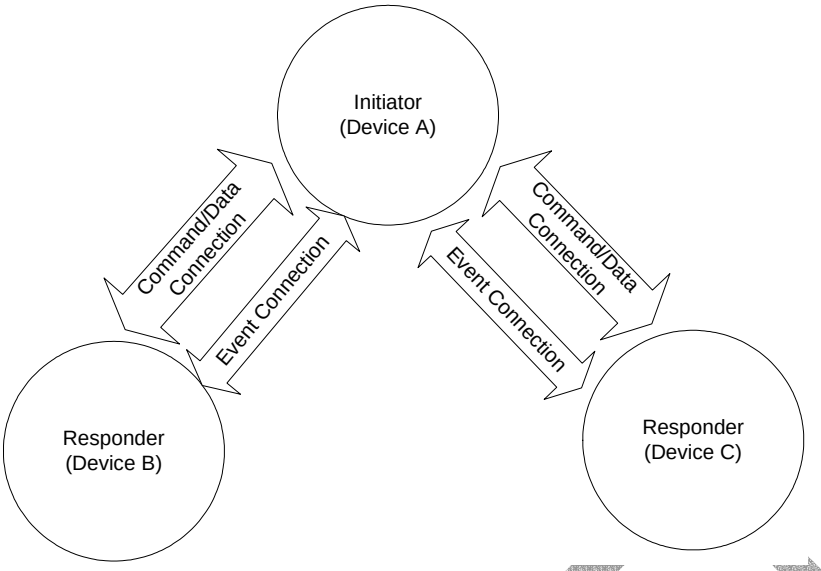


Figure 4: PTP-IP Communication Model

The Event packets are transported separately from the Operation and Data transaction packets because of their asynchronous nature.

2.2.1 Command/Data TCP connection

The PTP-IP Command/Data Connection is used for transporting PTP Operation Request, Response and Data transaction packets as well as PTP-IP specific packets.

This connection is established by the Initiator device and identified by the local and Responder's IP addresses and port numbers. The Responder's IP address and port number are usually determined through the device discovery mechanism.

2.2.2 Event TCP connection

The PTP-IP Event Connection is the second connection that the Initiator opens with the remote Responder. This connection is used for transporting PTP Event transaction packets as well as PTP-IP specific packets.

This connection is established by the Initiator device and identified by the local and remote IP addresses and port numbers. The Responder's IP address and port number are usually determined through the device discovery mechanism.

2.2.3 Transport Channel Management

2.2.3.1 Establishment of PTP-IP Connection

In the communication between image Initiator and Responder device, it's the responsibility of the Initiator device to initiate the establishment of PTP-IP TCP connections to the Responder device whenever these transport channels are needed.

As has been already stated earlier, the PTP-IP transport channels are implemented using TCP connections: one for Command/Data packet transport and one for Event packet transport. Each of these connections is identified by pairs of local and remote IP and TCP port numbers.

In the communication model, the Initiator's PTP-IP is a TCP client and the Responder's PTP-IP is a TCP server. The Responder waits for incoming connection requests on the predefined (or dynamically allocated) TCP port number (default value is 15740).

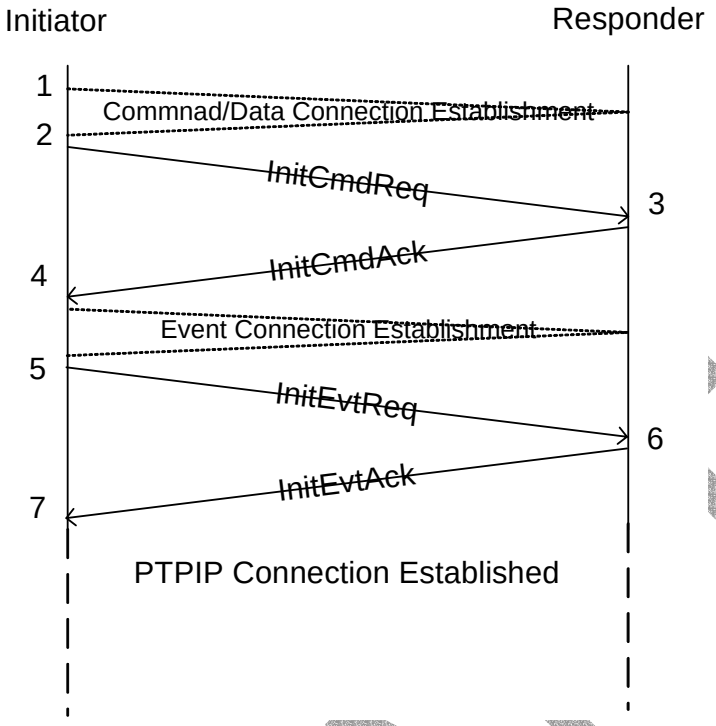


Figure 5: Connection Establishment Sequence

The TCP connections establishment SHALL take place in the sequence described in Figure 5 and presented in the following:

1. The Initiator initiates the establishment of the underlying Command/Data TCP connection with the Responder.
2. Immediately after this TCP connection is established, the Initiator sends the ***Init Command Request*** PTP-IP packet that contains its identity (GUID and Friendly Name).
3. The Responder can answer either with ***Init Command Ack*** PTP-IP Packet (letting the Initiator know that it can continue with the connection establishment procedure) or with an ***Init Fail*** PTP-IP packet (letting the Initiator know that access to its services are denied) and will close the Command/Data TCP connection. In the scenario where the Initiator receives the ***Init Fail*** PTP-IP packet, it SHALL close its end of the Command/Data TCP connection.
4. After the Initiator receives ***Init Command Ack*** packet in the previous phase, it SHALL initiate the Event TCP connection with the Responder.
5. Immediately after this TCP connection is established, the Initiator SHALL send the ***Init Event Request*** PTP-IP Packet that carries the previously assigned Connection Number. The Responder SHOULD use this number to associate the two TCP connections (Command/Data and Event) as belonging to same PTP-IP connection and to the same PTP session.
6. In response, the Responder SHALL send an ***Init Event Ack*** packet on the Event TCP connection. In the case the Responder cannot allocate enough resources, it SHALL send an ***Init Fail*** packet.

7. Once the Initiator receives the ***Init Event Ack*** packet, the PTP-IP connection is considered established and further data communication can take place.

In case the second TCP connection establishment fails, the Command/Data connection SHALL be closed. The Initiator MAY re-try to establish the PTP-IP transport channels later in time.

It is recommended that a timeout of 30 seconds be set during the PTP-IP connection establishment.

2.2.3.2 Configuration of Transport Channels

According to PTP Clause 7 [1], the PTP-IP layer requires reliable and error free transport channels. Such transport channels can be achieved by a proper configuration of the TCP connections.

The TCP connections SHOULD be created with the following options:

- Keep Alive option set on both Initiator and Responder - this option SHOULD be enabled in order for the TCP entities on both the Initiator and the Responder to keep the connection alive during inactivity periods. If this option is enabled, then the two TCP stacks will check, in a TCP protocol specific way if the peer device is still responding. This is done in a transparent way from the PTP-IP layer point of view.
- Disable Nagle's algorithm on both the Initiator and the Responder TCP/IP stacks (in order to avoid unnecessary delays in transporting the command codes and responses).

2.2.3.3 Shutdown of PTP-IP Connection

In the communication between the Initiator and the Responder, it is the Initiator that initiates the shutdown of the connection channels whenever these channels are not needed anymore.

In case that either the Initiator or the Responder has encountered an error on the open transport channels which discredited the reliability of those channels, it is a reason to initiate the PTP-IP connection shutdown procedure.

Both PTP-IP TCP connections (Command/Data and Event) MUST be closed during the shutdown procedure.

2.3 Transfer Packet Types

This section describes a set of signalling packet types used during communication between the Initiator and the Responder PTP-IP entities. PTP-IP packet types correspond to the Operation Requests, Response, Data and Event transactions defined in PTP protocol.

All multi-byte values in PTP-IP packets are represented using the little-endian format.

The general format of PTP-IP packet is shown in Figure 6.

Field	Size [Bytes]	Data Type
Length	4	UINT32
Packet Type	4	UINT32
Payload	?	?

Figure 6: General Format of PTP-IP Packet

Length: 4 Bytes

Length parameter determines the total size of the packet, including the header.

Packet Type : 4 Bytes

This parameter identifies the type of packet. The table below shows the valid values of this parameter. 0x0000NNNN* means hidden Value.

- 0x00000000 Invalid Value
- 0x0000NNNN* Init Command Request Packet
- 0x0000NNNN* Init Command Ack
- 0x0000NNNN* Init Event Request Packet
- 0x0000NNNN* Init Event Ack Packet
- 0x0000NNNN* Init Fail Packet
- 0x0000NNNN* Operation Request Packet
- 0x0000NNNN* Operation Response Packet
- 0x0000NNNN* Event Packet
- 0x0000NNNN* Start Data Packet
- 0x0000NNNN* Data Packet
- 0x0000NNNN* Cancel Packet
- 0x0000NNNN* End Data Packet
- 0x0000NNNN* Probe Request Packet
- 0x0000NNNN* Probe Response Packet
- 0x0000NNNN* – 0xFFFFFFFF Reserved

Payload

Carries either the higher layer data or transport specific data.

2.3.1 Init Command Request Packet

This packet is used immediately after the Command/Data TCP transport channel is established. The transfer direction of this packet is from the Initiator to the Responder. It is transmitted on Data/Command TCP connection and it is used to signal the Responder the identity of the Initiator, in case the Responder implements a filtering mechanism.

The structure of ***Init Packet*** is shown in Figure 7.

Field	Size [Bytes]	Data Type
Length	4	UINT32
Packet Type	4	UINT32
Initiator GUID	16	UINT8
Initiator Friendly Name	Variable	UINT16
Initiator Protocol Version	4	UINT32

Figure 7: Init Command Request PTP-IP Packet

Initiator GUID: 16 Bytes

It represents a globally unique identifier (GUID) of the Initiator.

Initiator Friendly Name: Variable

It is a Unicode string containing a description of the Initiator. This string MAY be used in the user interface of the Responder (to confirm or reject connection requests from a given Initiator).

Initiator Protocol Version: 4 Bytes

Number that represents the protocol version implemented by the Initiator. This field is made out of a major number (16 most significant bits) and a minor number (16 least significant bits). This field should be set to the BINARY PROTOCOL VERSION defined in section 3.

Note:

- *The Initiator GUID, together with the Responder GUID can be used for device bonding. The Initiator's GUID consisting of all bytes set to 0xFF represent a special "anonymous" value. It is up to the Responder to accept connections from such Initiators. For more explanations, please see annex 5.3.*
- The Initiator Friendly name is null-terminated (ending with a NULL UNICODE character). Its maximum size is 40 Unicode characters including the NULL Unicode character at the end. If this value is not used the field MUST contain one NULL UNICODE character.

2.3.2 Init Command ACK Packet

This packet is used by the Responder in response to an ***Init Command Request Packet,*** to assign the Connection Number for the PTP-IP session. The transfer direction of this packet is from the Responder to the Initiator. It is transmitted on Data/Command TCP connection.

The ***Init Command ACK Packet*** is shown in Figure 8.

Field	Size [Bytes]	Data Type
Length	4	UINT32
Packet Type	4	UINT32
Connection Number	4	UINT32
Responder GUID	16	UINT8
Responder Friendly Name	Variable	UINT16
Responder Protocol Version	4	UINT32

Figure 8: Init Command Ack PTP-IP Packet

Connection Number: 4 Bytes

It is a unique number generated by the Responder that is used to associate the TCP transport channels belonging to same PTP session.

Responder GUID: 16 Bytes

It represents a globally unique identifier (GUID) of the Responder.

Responder Friendly Name: Variable

It is a UNICODE string containing a description of the Responder. This string MAY be used in the user interface of the Initiator.

Responder Protocol Version: 4 Bytes

Number that represents the protocol version implemented by the Responder. This field is made out of a major number (16 most significant bits) and a minor number (16 least significant bits). If the Initiator accepts the version received from the Responder, then it should continue the PTP session. If the Initiator doesn't support the Responder's version, then it should close the connection. This field should be set to the BINARY PROTOCOL VERSION defined in section 3.

Note:

- The Responder Friendly name is null-terminated (ending with a NULL UNICODE character). Its maximum size is 40 Unicode characters including the NULL Unicode character at the end. If this value is not used this field MUST contain one NULL UNICODE character.
- The Connection Number remains valid during the Connection Establishment Sequence between the Initiator and the Responder. Once the Responder has successfully associated the two TCP connections belonging to same PTP-IP connection, the Connection Number can be reused.

2.3.3 Init Event Request Packet

This packet is used in order to establish the Event TCP Connection after the Command/Data TCP Connection is established. When the Initiator receives a valid ***Init Command Ack Packet*** it establishes the Event TCP connection and transmits this packet. The connection number received via ***Init Command Ack Packet*** is transported in this packet. The transfer direction of this packet is from Initiator to the Responder. It is transmitted on the Event TCP connection.

The ***Init Event Request Packet*** is shown in Figure 9.

Field	Size [Bytes]	Data Type
Length	4	UINT32
Packet Type	4	UINT32
Connection Number	4	UINT32

Figure 9: Init Event Request PTP-IP Packet

Connection Number: 4 Bytes

It is a 32-bit value that has been received in ***Init Command Ack Packet***.

2.3.4 Init Event Ack Packet

This packet is used in PTP-IP by the Responder to inform the Initiator that the PTP-IP connection establishment completed successfully. The transfer direction of this packet is from the Responder to the Initiator. It is transmitted on the Event TCP connection.

The ***Init Event Ack Packet*** is shown in Figure 10

Field	Size [Bytes]	Data Type
Length	4	UINT32
Packet Type	4	UINT32

Figure 10: Init Event Ack PTP-IP Packet

2.3.5 Init Fail Packet

This packet is used in PTP-IP by the Responder to inform the Initiator that the PTP-IP connection establishment failed. The reason of failure is reported in the Reason field. Upon receiving the packet, the Initiator MUST close the Command/Data Connection with the Responder that rejects the Event Connection request. After issuing an ***Init Fail Packet***, the Responder SHALL close the PTP-IP connection (TCP connections initiated from the Initiator that has been rejected). The ***Init Fail Packet*** can be transported on either of TCP connections. For more details on the PTP-IP connection establishment see section Establishment of PTP-IP Connection.

The ***Init Fail Packet*** is shown in Figure 11.

Field	Size [Bytes]	Data Type
Length	4	UINT32
Packet Type	4	UINT32
Reason	4	UINT32

Figure 11: Init Fail PTP-IP Packet

Reason field contains the rejection code. The following codes are defined:

- 0x0000NNNN* – FAIL_REJECTED_INITIATOR. Means that the Responder implements a device bonding mechanism and the Initiator requesting the connection is not one of the "allowed" devices. See annex 5.3 for more details on bonding mechanisms.
- 0x0000NNNN* – FAIL_BUSY. Means that the Responder has too many active connections. The Initiator MAY try to establish a connection latter.
- 0x0000NNNN* – FAIL_UNSPECIFIED. Covers all other rejection cases.

2.3.6 Operation Request Packet

This packet is used in PTP-IP to transport PTP operation requests defined in PTP Clause 9.3.2 [1]. PTP-IP ***Operation Request Packets*** are issued by the Initiator and are transported to the Responder device via the PTP-IP Command/Data transport channel. The direction of this packet is from Initiator to Responder.

The format of the ***Operation Request Packet*** is shown in Figure 12.

Field	Size [Bytes]	Data Type
Length	4	UINT32
Packet Type	4	UINT32
Data Phase Info	4	UINT32
Operation Code	2	UINT16
TransactionID	4	UINT32
Parameter 1	4	UINT32
Parameter 2	4	UINT32
Parameter 3	4	UINT32
Parameter 4	4	UINT32
Parameter 5	4	UINT32

Figure 12: Operation Request Packet Format

Data Phase Info : 4 Bytes

Indication whether the next data phase is a data out phase or a data in/no data phase.

Defined values are as follows:

- 0x0000NNNN* – No Data or Data In Phase (as the Initiator doesn't know if the remote Responder will provide some data or a response)
- 0x0000NNNN* – Data Out Phase
- 0x0000NNNN* – Unknown Data Phase (please note implementation comment)[1]

[1] Important implementation note: The use of the "Unknown Data Phase" value in the ***Operation Request Packet*** may significantly complicate the implementation of the protocol. When a packet with such value is issued by the Initiator neither side of the communication "knows" what type of packet to expect next, and, more importantly, in what direction. The knowledge may be available at higher levels of the communication stack (PTP or application) or may not be available at all until the next packet arrives for transmission at a later time. The latter is an important case of "bridge" or "repeater" type of application that translate PTP communication between USB and PTP-IP channels.
In case of implementations for devices with limited processing capabilities it may not always be possible to fully support this feature. In those cases, if the Responder cannot handle the

In the case of a standard PTP operation request, this field MUST always be set by Initiator to either “No Data or Data In Phase” or “Data Out Phase”.

In the case of a vendor specific PTP operation request:

- if the next data phase is known, this field MUST always be set by Initiator to either “No Data or Data In Phase” or “Data Out Phase”.
- If the next data phase is unknown (in exceptional cases, involving certain type of specialized bridging applications), this field MAY be set to “Unknown Data Phase”.

Operation Code : 2 Bytes

Contains the PTP Operation Code, defined in the PTP Clause 10.2 & 10.3 [1].

TransactionID : 4 Bytes

Contains the TransactionID, which is unique in the context of an open session on a connection to the remote device, as defined in the PTP Clause 9.3.1 [1]. TransactionIDs are continuous sequential numbers starting from 0x00000001. A special TransactionID value of 0x00000000 MAY only be used in the OpenSession or GetDeviceInfo Operations Request. A special value of 0xFFFFFFFF MUST be used every time when this parameter is not applicable for the packet.

Parameter 1-5: 4 Bytes

Those fields hold operation-specific parameters. The interpretation of those parameters depends on the Operation Code parameter. Operations MAY hold up to 5 parameters. The usage is defined in the PTP Clause 10.1 & 10.4 [1].

Note:

- If “Data Phase Info” field is set to “Data Out Phase”, then this packet MUST be followed by a ***Start Data Packet***.
- If the Initiator wants to transfer a null data object to the Responder, than it has two options:
    - Set the “Data Phase Info” field to “No data or data in phase”, in which case the responder will follow up with an ***Operation Response Packet***, without waiting for a data
    - Set the “Data Phase Info” field to “Data Out Phase”. In this case, the data out phase MUST consist of exactly one ***Start Data Packet***, having the “Total Data Length” field set to 0x00000000, and one empty ***End Data Packet*** The Initiator MUST not send any consequent ***Data Packets***.

2.3.7 Operation Response Packet

This packet is used in PTP-IP to transport Operation Responses defined in PTP Clause 9.3.4 [1]. PTP-IP ***Operation Response Packets*** are issued by the Responder and transported to the Initiator via PTP-IP Command/Data connection. PTP-IP ***Operation Response Packets*** are only issued by the Responder to indicate that the requested operation transaction has been completed and to pass the operation result.

The format of the ***Operation Response Packet*** is shown in Figure 13

Field	Size [Bytes]	Data Type
Length	4	UINT32

"Unknown Data Phase" value properly, it is recommended that it closes the connection upon receipt of such packets.

This document does not address the interface (API) issues. Different approaches to the design of the API may also affect the complexity of the implementation that fully supports this feature.

Packet Type	4	UINT32
Response Code	2	UINT16
TransactionID	4	UINT32
Parameter 1	4	UINT32
Parameter 2	4	UINT32
Parameter 3	4	UINT32
Parameter 4	4	UINT32
Parameter 5	4	UINT32

Figure 13: Operation Response Packet Format

Response Code : 2 Bytes
Contains the PTP Operation Response code, defined in the PTP Clause 11.1 & 11.3 [1].

TransactionID : 4 Bytes

Contains the TransactionID, which is unique within an open session on a connection to the remote device, as defined in the PTP Clause 9.3.1 [1]. TransactionIDs are continuous sequential numbers starting from 0x00000001. A special TransactionID value of 0x00000000 MAY only be used in the OpenSession or GetDeviceInfo Operations Request. A special value of 0xFFFFFFFF MUST be used every time when this parameter is not applicable for the current packet.

Parameter 1-5: 4 Bytes

Those fields hold operation-specific parameters. The interpretation of those parameters depends on the operation for which the response has been generated as well as the particular Response Code value. ***Operation Response Packet*** MAY hold up to 5 parameters. The usage is defined in the PTP Clause 9.3.4 [1].

2.3.8 Event Packet

This packet is used to transport Events defined in PTP clause 12.2 [1]. The events are used to inform Initiator about the Responder state change. The direction of this packet is from the Responder to the Initiator. It is transmitted on the Event TCP connection.

The format of the ***Event Packet*** is shown in Figure 14.

Field	Size [Bytes]	Data Type
Length	4	UINT32
Packet Type	4	UINT32
Event Code	2	UINT16
TransactionID	4	UINT32
Parameter 1	4	UINT32
Parameter 2	4	UINT32
Parameter 3	4	UINT32

Figure 14: Event Packet Format

Event Code : 2 Bytes

Contains the Event Code, defined in the PTP Clause 12.3 & 12.4 [1].

TransactionID : 4 Bytes

Contains the TransactionID, defined in the PTP Clause 9.3.1 [1]. If the event corresponds to a previously initiated transaction, this field SHALL hold the TransactionID of that operation. If the event is not specific to a particular transaction, this field SHALL be set to 0xFFFFFFFF.

Parameter 1-3 : 4 Bytes

Those field hold event-specific parameters. The interpretation of those parameters depends on the Event Code. ***Event Packet*** may hold up to 3 parameters. The usage is defined in the PTP Clause 12.5 [1].

2.3.9 Start Data Packet

This type of packet is used in PTP-IP to signal the beginning of a data transfer. This packet is bi-directional, so this packet is either from the Responder to the Initiator or from the Initiator to the Responder. It is transmitted on Command/Data TCP connection.

The packet layout for ***Start Data Packet*** is presented in Figure 15.

Field	Size [Bytes]	Data Type
Length	4	UINT32
Packet Type	4	UINT32
TransactionID	4	UINT32
Total Data Length	8	UINT64

Figure 15: Start Data Packet

TransactionID : 4 Bytes

Contains the TransactionID, defined in the PTP Clause 9.3.1 [1].

Total Data Length: 8 Bytes

Contains the size of data that is going to be transferred during the data-in or data-out phase. A value of 0xFFFFFFFFFFFFFFFF indicates that the size of the data is not known at the beginning of the data phase. The usage of this value should be limited to the cases where the transferred object is of an unknown size (for example: streaming data).

2.3.10 Data Packet

This packet is used in PTP-IP to transport data. ***Data Packets*** are only used during data phase of a transaction and can be issued either by the Initiator or Responder in the direction of the data flow: for the data-in phase – from the Responder to the Initiator; for the data-out phase – from the Initiator to the Responder. ***Data Packets*** are transmitted on Command/Data TCP connection.

Since the TCP transport is an error-free, stream-based protocol, normally there is no need in doing fragmentation and assembly of large data packets. However, a basic fragmentation mechanism MAY be utilized to allow for a simple data transfer cancelling mechanism. No error checking is required.

The format of the ***Data Packet*** is shown in Figure 16

Field	Size [Bytes]	Data Type
Length	4	UINT32
Packet Type	4	UINT32
TransactionID	4	UINT32
Data Payload	?	UINT8

Figure 16: Data Packet

TransactionID : 4 Bytes

Contains TransactionID, defined in the PTP Clause 9.3.1 [1].

Data Payload: variable size

Contains the data payload of the packet.

2.3.11 End Data Packet

This packet is used in PTP-IP to indicate the end of the data phase. ***End Data Packet*** can also carry useful data. This packet is only used during data phase of a transaction and can be issued either by the Initiator or Responder in the direction of the data flow: for the data-in phase – from the Responder to the Initiator; for the data-out phase – from the Initiator to the Responder.

The format of the ***End Data Packet*** is shown in Figure 17

Field	Size [Bytes]	Data Type
Length	4	UINT32
Packet Type	4	UINT32
TransactionID	4	UINT32
Data Payload	?	UINT8

Figure 17: End Data Packet

TransactionID : 4 Bytes

Contains TransactionID, defined in the PTP Clause 9.3.1 [1].

Data Payload: variable size

Contains the data payload of the packet, if any.

2.3.12 Cancel Packet

This packet is used by the transport to cancel a transaction. Its format is presented in Figure 18.

Field	Size [Bytes]	Data Type
Length	4	UINT32
Packet Type	4	UINT32
TransactionID	4	UINT32

Figure 18: Cancel Packet

TransactionID : 4 Bytes

Contains TransactionID, defined in the PTP Clause 9.3.1 [1].

The direction of this packet is from Initiator to the Responder. It is transmitted on the Command/Data TCP connection and on the Event TCP connection.

2.3.13 Probe Request Packet

This packet can be used in PTP-IP by both Initiator and Responder to check if peer device is still active. Upon receiving such packet, the device MUST respond immediately with a ***Probe Response Packet***. If no response is received within a reasonable period of time, the device initiating this check will close the active PTP-IP session(s) with the remote device.

The ***Probe Request Packet*** is shown in Figure 19 and it is sent on the Event TCP connection.

Field	Size [Bytes]	Data Type
Length	4	UINT32
Packet Type	4	UINT32

Figure 19: Probe Request Packet

Note: This packet should be used with utmost care in order to avoid overloading of the LAN.

- Initiator to Responder - it is recommended that this packet is used only during a PTP transaction (e.g. when a format command is issued, if the storage media is large, the response time can be quite large), in order to check out if the Responder is still active or not.
- Responder to Initiator – it is recommended to use this packet only when the Responder receives a request for a new PTP-IP session while one ore more other sessions are active. In this case, the Responder can check if the existing PTP-IP connections are still active.
- It is recommended that a timeout of 10 seconds be set between sending the ***Probe Request Packet*** and receiving the ***Probe Response Packet***.

2.3.14 Probe Response Packet

This packet can be used in PTP-IP by both Initiator and Responder, as a response to a ***Probe Request Packet***. Upon receiving a ***Probe Request Packet***, a ***Probe Response Packet*** MUST be issued immediately.

The ***Probe Response Packet*** is shown in Figure 20 and it is sent on the Event TCP connection.

Field	Size [Bytes]	Data Type
Length	4	UINT32
Packet Type	4	UINT32

Figure 20: Probe Response Packet

Note: This packet MUST not be issued unless a ***Probe Request Packet*** is received.

2.4 Session Implementation

A new session request from an Initiator in the PTP layer will generate the creation of two new TCP connections that correspond to that session at the PTP-IP layer. The TCP connections are enough at the PTP-IP layer to uniquely identify a session; therefore there is no need to transport the SessionID as a separate value from the Initiator to the Responder.

2.5 Device Disconnection or Network Loss Handling

In PTP-IP, the device disconnection or network loss detection is based on a number of criteria:

- The capabilities of the network layer to notify the applications about a network disconnect notification (e.g. media disconnected).
- Loosing the connection on one of the PTP-IP channels (Data/Commands or Event connections) (e.g. broken socket).
- Transmission timeout on one of the PTP-IP channels

2.6 Data Flow Control

In order to prevent a device from overloading another device with data during the data phase of a transaction, usually, a data flow control mechanism is implemented. However, there is no need for this to be implemented in the PTP-IP, since the underlying TCP layer performs the flow control operation.

TCP layer implements flow control at both levels: receiving device and network, avoiding flooding with data either a slow receiver or a slow network. Therefore, by using TCP connections as communication channels in PTP-IP, the flow control issues are solved.

2.7 Transaction Cancelling Mechanism

There are two possible cases of cancelling a transaction: Initiator initiates the cancel or Responder initiates the cancel.

2.7.1 Initiator Generated Cancel

2.7.1.1 Data-Out Phase

In order for the Initiator to cancel an ongoing data-out transfer it MUST issue a ***Cancel Packet*** on the Data/Command and the Event TCP connections, and cease any data transfer on the Data/Command connection. When the Responder receives the ***Cancel Packet*** on either Data/Command or the Event TCP connection it MUST as soon as possible interrupt and cancel the ongoing transaction, read and discard all the remaining ***Data Packets*** pertaining to the current transaction (if any), and issue an ***Operation Response Packet*** with the Transaction_Cancelled response code.

2.7.1.2 Data-In Phase

In order for the Initiator to cancel an ongoing data-in transfer, it MUST issue a ***Cancel Packet*** on the Data/Command and the Event TCP connections. When the Responder receives the ***Cancel Packet*** on either Data/Command or the Event TCP connection it MUST as soon as possible interrupt and cancel the ongoing transaction and issue an ***Operation Response Packet*** with the Transaction_Cancelled response code.

2.7.1.3 Other Transaction Phases

The Initiator MAY cancel a transaction while in any transaction phase by sending a ***Cancel Packet*** on the Data/Command and the Event TCP Connection. If the cancellation refers to the current transaction, the Responder MUST issue an ***Operation Response Packet*** with the Transaction_Cancelled response code.

2.7.2 Responder Generated Cancel

2.7.2.1 Data-Out Phase

In order for the Responder to cancel an ongoing data-out phase, it MUST issue an ***Operation Response Packet*** with the Transaction_Cancelled response code, or any other error response code that indicates the reason for data transfer interruption. The Responder then MUST read and discard all the ***Data Packets*** from the Data/Command connections pertaining to the current transaction.

2.7.2.2 Data-In Phase

In order for the Responder to cancel an ongoing data-in phase, it MUST issue an ***Operation Response Packet*** on the Data/Command TCP connection, with the Transaction_Cancelled response code or any other error response code that indicates the reason for data transfer interruption.

2.7.2.3 Other Transaction Phases

Responder MAY cancel a transaction in any phase by sending an ***Operation Response Packet*** with the Transaction_Cancelled response code or any other response code that indicates the reason for transaction interruption.

2.7.3 Asynchronous Operation Cancelling Mechanism

In order for the Initiator to cancel an asynchronous operation it MUST send a ***Cancel Packet*** containing the TransactionID of the asynchronous operation (e.g. InitiateCapture) on both Data/Command and the Event TCP Connections. It MAY send those packets at any time.

The Responder will end the operation with a PTP event (e.g. Capture Complete Event). The logic of interpreting this cancel request is the responsibility of the application layer of the Responder.

3 Protocol Version

The BINARY PROTOCOL VERSION of this specification is 0x00010000 (Revision 1.0). The binary protocol version is made out of a major number (16 most significant bits) and a minor number (16 least significant bits). This number will increase to indicate a newer protocol version, in the following manner:

- The minor version is incremented to reflect minor changes in the protocol specifications. The devices implementing a new minor version have to support in full all the lesser minor versions of the same major version.
- The major version is incremented to reflect major changes in the protocol specifications. The new protocol specification may not be compatible with older version of the specification having a smaller major number.

4 PTP-IP Port

If no device discovery protocol is implemented, each Responder and Initiator SHOULD initialize to use the port number as described below.

Port Name	Port No.	Type	Description
PTP-IP service	15740	TCP	Default port on which the Responder waits for incoming TCP connections. The port is approved by IANA.

5 Informative Annex

5.1 Address Configuration

The foundation for PTP-IP is the TCP/IP protocol and the key to this protocol is the addressing mechanism. Both the Responder and the Initiator will obtain valid IP addresses. There are a few methods of obtaining valid IP addresses:

- Manual configuration – the IP address and other networking attributes are configured manually by the user, to reflect the topology and address schema of the local area network in which the imaging devices will function;
- DHCP-based configuration –the imaging devices should implement a DHCP client that will automatically obtain an IP address from the DHCP server running in the network;
- Dynamic Configuration of IPv4 Link-Local Addresses (v4LL) – a standard that describes a mechanism for an automatic self-configuration of networking devices on a TCP/IP-based LAN, without having to setup a DHCP server.

The following procedure is suggested in order to deal with the device configuration of TCP/IP attributes:

- If the attributes are manually configured – that configuration is to be used;
- Otherwise the device should use DHCP protocol to obtain an IP address and other configuration parameters (therefore, the device should implement a DHCP client);
- If no DHCP server is present in the network, the Dynamic Configuration of v4LL should be employed.

5.2 Device Discovery and Enumeration

In PTP-IP the Initiator can be configured with the address of the Responder to connect to in a few ways:

- Manual configuration – the Initiator will be manually configured with the IP address or a list of IP addresses for the Responders it can connect to. The Initiator will try a PTP-IP connection on the indicated IP address. If the connection is successful than the Responder is present and a PTP session can be established.
- Zeroconf – A family of protocols and recommendations providing an automatic configuration, device and service discovery, defined by IETF Zeroconf Working Group. [4]
- UPnP –The family of protocols defined by UPnP Forum [3] to facilitate automatic device configuration, service discovery and invocation. Only the device and service discovery features of UPnP are to be used.
- Any other discovery mechanism

The PTP specification, clause 7.4, requires support for device discovery and enumeration from its transports. Therefore, the PTP-IP layer should use an implementation of such service.

Despite what service discovery and enumeration is used, we recommend that the device GUID will be part of the announcement packets, so the Initiator will be able to filter (if it wants) specific devices, and generate selective notifications.

5.3 Device Bonding and Authentication

Authentication is not the responsibility of PTP-IP layer. If authentication is required, it should rely on the solutions available in the underlying network (physical/data-link layers).

Bonding can rely on either mechanisms available in the underlying network or can be implemented at the application layer, based on mechanisms provided by the PTP-IP layer, as follows.

Each device in PTP-IP context SHOULD have a globally unique identifier - GUID (16 bytes unique number) that may be used for the "device bonding". For those device that support this feature the PTP-IP layer provides a mechanism which allows the Initiator and Responder to exchange their GUIDs. This, in turn, allows the application layer to implement device-level access control (e.g. a new device arrives in the network and it is discovered; the device can accept selective connections from known Initiators only, or it can initiate connections to only known Responders).

Regardless of what service discovery and enumeration methods are used, it is recommended that the device GUID be part of the announcement packets, so the *Initiator* is able to filter out (if desired) specific devices, and generate selective notifications.

5.4 Data Security

This section is not within the scope of PTP-IP protocol. Data security will rely on the solutions available in the underlying network (physical/data-link layers).

5.5 Protocol Versions Interoperability

The protocol version numbering scheme is intended to provide the interoperability between the implementations of different versions of the protocol that have a common major number.

The following are the implementation guidelines that are intended to maximize the level of interoperability between the devices implementing different version of the protocol specification. In the following text “IPV” stands for “Initiator Protocol Version” and “RPV” stands for “Responder Protocol Version”.

1. Any device that implements IPV “A.B” (major “A”, minor “B”) should also support any IPV “A.X” where “X” is no greater than “B”.
2. Any device that implements RPV “A.B” (major “A”, minor “B”) should also support any RPV “A.X” where “X” is no greater than “B”.
3. Initiator should set its IPV to the highest supported level unless the Responder is known to implement a lesser version (for example, from a discovery process), in which case set the value of IPV equal to that of RPV.
4. Responder should set its RPV to the highest supported level unless the value of the IPV is smaller, in which case set the value of RPV equal to that of IPV.

By following these rules, the devices are guaranteed to establish the communication on the highest level of the protocol specification supported by both sides.

The versions of the protocol specification that differ in the major numbers are not assumed to be compatible.

5.6 PTPIP Cancellation Examples

This section presents several cancellation examples.

5.6.1 Initiator Generated Cancel

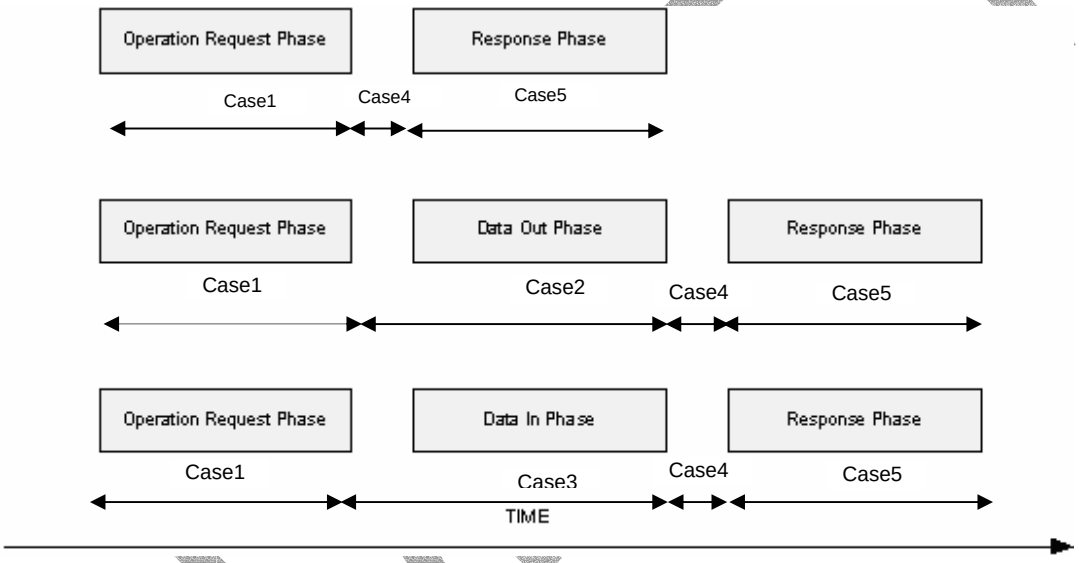
There are two types of cancels in PTP:

1. synchronous cancel (when the current transaction is canceled)

2. asynchronous cancel (when the Initiator initiated an asynchronous operation, and wants to cancel it using its transaction ID).

In the case of asynchronous cancel, the Responder should terminate the asynchronous operation that is indicated by the transaction ID.

In the case of synchronous transaction, the transaction ID in the cancel packet must be equal to the current transaction ID. Following cases describe the synchronous canceling mechanism from the Responder point of view and refer to the Responder time scale.



Case 1- Cancel packet is received before the Operation Request Packet.

Subcase 1.1 Cancel packet received first via Commad/Data connection.

This case is not possible.

Subcase 1.2 Cancel packet received first via Event connection.

The Responder ignores the Cancel Packet.

Case 2 - Cancel packet is received after Operation Request Packet that can be followed by data-out or when data-out has started but not completely received by Responder.

Subcase 2.1 Cancel packet received first via Command/Data connection

If there is any processing for the current operation then the Responder terminates the processing.

The Responder sends the Operation Response Packet with the response code "Transaction Canceled".

The Responder ignores the Cancel Packet received via Event connection.

Subcase 2.2 Cancel packet received first via Event connection

If there is any processing for the current operation then the Responder terminates the processing.

The Responder waits for the Cancel Packet via Command/Data connection and ignores Start Data, Data and End Data packets received via Command/Data connection.

The Responder sends the Operation Response Packet with the response code "Transaction Canceled".

Case 3 - Cancel packet is received after Responder has begun but before it has finished sending data-in.

Subcase 3.1 Cancel packet received first via Command/Data connection

If there is any processing for the current operation then the Responder terminates the processing.

The Responder stops sending data-in packets.

The Responder sends the Operation Response Packet with the response code "Transaction Canceled".

The Responder ignores the Cancel Packet received via Event connection.

Subcase 3.2 Cancel packet received first via Event connection

If there is any processing for the current operation then the Responder terminates the processing.

The Responder stops sending data-in packets.

The Responder waits for the Cancel Packet via Command/Data connection.

The Responder sends the Operation Response Packet with the response code "Transaction Canceled".

Case 4 - Cancel packet is received after Responder has finished receiving data-out or sending data-in (if any) but before it has started sending Operation Response Packet.

Subcase 4.1 Cancel packet received first via Command/Data connection

If there is any processing for the current operation then the Responder terminates the processing.

The Responder sends the Operation Response Packet with the response code "Transaction Canceled".

The Responder ignores the Cancel Packet received via Event connection.

Subcase 4.2 Cancel packet received first via Event connection

If there is any processing for the current operation then the Responder terminates the processing.

The Responder waits for the Cancel Packet via Command/Data connection.

The Responder sends the Operation Response Packet with the response code "Transaction Canceled".

Case 5 - Cancel packet is received after Responder has begun sending the Response Packet

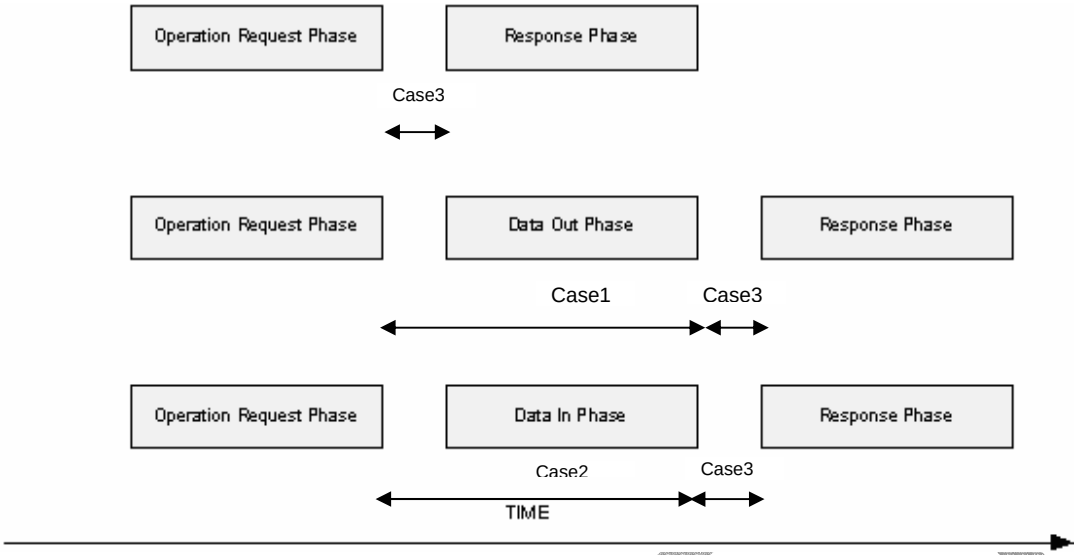
Subcase 5.1 Cancel packet received first via Command/Data connection.

The Responder ignores the Cancel Packet.

Subcase 5.2 Cancel packet received first via Event connection.

The Responder ignores the Cancel Packet.

5.6.2 Responder Generated Cancel



Case 1 - Responder cancels right after receiving Operation Request Packet that can be followed by data-out or when data-out has started but not completely sent by Initiator

The Responder sends the Operation Response Packet with the response code set to the appropriate reason of error.
The Responder ignores all following data-out packets.
The Initiator receives the Operation Response Packet.
The Initiator terminates sending data-out packets.

Case 2 - Responder cancels when the data-in is in progress

The Responder stops sending data-in packets.
The Responder sends the Operation Response Packet with the response code set to the appropriate reason of error.
The Initiator receives the Operation Response Packet.
The Initiator discards the received data-in.

Case 3 - Responder cancels after data-out or data-in is complete, or when there is no data phase

The Responder sends the Operation Response Packet with the response code set to the appropriate reason of error.
The Initiator receives the Operation Response Packet.

The standards of the Camera & Imaging Products Association are drawn up with no representation made as to the relationship of the standards to intellectual properties (patents, utility patents, etc.).

The Camera & Imaging Products Association shall bear no responsibility for any intellectual property rights concerning the contents of this standard.

CIPA DC-X005-2005

Published on 28 July, 2005

Published by Camera & Imaging Products Association
JCII BLDG., 25, Ichiban-cho, Chiyoda-ku, Tokyo, 102-0082 Japan
TEL +81-3-5276-3891 FAX +81-3-5276-3893