Video Services Forum (VSF)
Technical Recommendation TR-10-14

Internet Protocol Media Experience (IPMX):

IPMX USB

September 24, 2024

INTELLECTUAL PROPERTY RIGHTS

RECIPIENTS OF THIS DOCUMENT ARE REQUESTED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT CLAIMS OR OTHER INTELLECTUAL PROPERTY RIGHTS OF WHICH THEY MAY BE AWARE THAT MIGHT BE INFRINGED BY ANY IMPLEMENTATION OF THE RECOMMENDATION SET FORTH IN THIS DOCUMENT, AND TO PROVIDE SUPPORTING DOCUMENTATION.

THIS RECOMMENDATION IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NONINFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY USE OF THIS RECOMMENDATION SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE FORUM, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER, DIRECTLY OR INDIRECTLY, ARISING FROM THE USE OF THIS RECOMMENDATION.

LIMITATION OF LIABILITY

VSF SHALL NOT BE LIABLE FOR ANY AND ALL DAMAGES, DIRECT OR INDIRECT, ARISING FROM OR RELATING TO ANY USE OF THE CONTENTS CONTAINED HEREIN, INCLUDING WITHOUT LIMITATION ANY AND ALL INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES (INCLUDING DAMAGES FOR LOSS OF BUSINESS, LOSS OF PROFITS, LITIGATION, OR THE LIKE), WHETHER BASED UPON BREACH OF CONTRACT, BREACH OF WARRANTY, TORT (INCLUDING NEGLIGENCE), PRODUCT LIABILITY OR OTHERWISE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE FOREGOING NEGATION OF DAMAGES IS A FUNDAMENTAL ELEMENT OF THE USE OF THE CONTENTS HEREOF, AND THESE CONTENTS WOULD NOT BE PUBLISHED BY VSF WITHOUT SUCH LIMITATIONS.

Executive Summary

IPMX was created to foster the adoption of open standards based protocols for interoperability over IP in the media and entertainment (M&E) and professional audio/video industries. This technical recommendation defines the transport layer technologies for IPMX USB-over-IP networks. The NMOS REST APIs from AMWA provides discovery, connection management, and control.

1 Introduction (Informative)

The main purpose of the IPMX USB is to enable low-latency management and control of USB communication between a Host and a remotely located peripheral as shown in Figure 1. Such remote peripheral might be of any type.

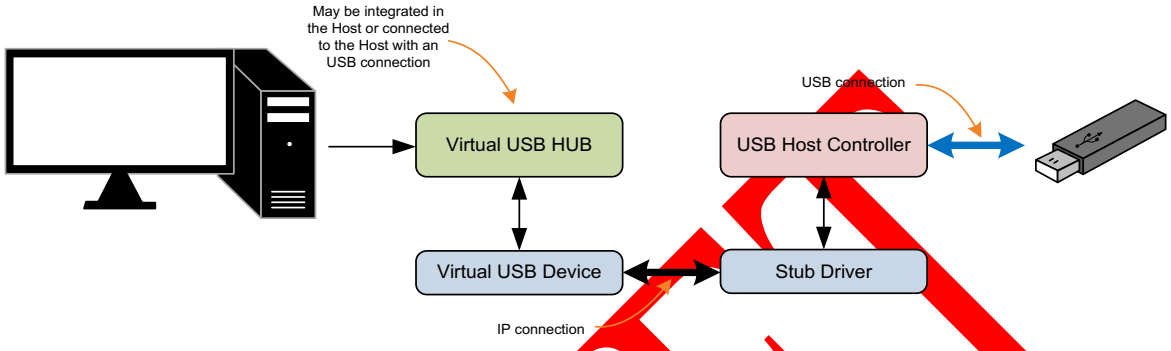

Figure 1: IPMX USB IP usage

The IPMX USB-over-IP-networks specification intends to provide USB streaming solutions to any peripheral type and any USB use-case, to serve the system vendor with flexibility to solve any use-case. Remote access to a peripheral's USB port might have near-real-time needs, to change a device state, or enable the Host responding quickly to some extreme events that occur on the peripheral side. Within a network where the bandwidth is not oversubscribed, a time latency of tens of Microseconds should be achieved.

IPMX USB is based on USB 2.0 specifications. All features of the USB 2.0 specification are supported, and multiple devices can be attached. All IPMX USB communications on the IP network are encrypted.

1.1 Contributors

The following individuals participated in the Video Services Forum IPMX working group that developed this technical recommendation.

Alain Bouchard (Matrox)	François-Pierre CLOUET (intoPIX)	Nick Nicas (AT&T)
Andre Testa (Matrox)	Jack Douglass (PacketStorm)	Paulo Francisco (Evertz)
Andreas Hildebrand (ALC NetworX)	Jean Lapierre (Matrox)	Raymond Hermans (Adeas)
Andrew Starks (Macnica)	Jean-Jacques Ostiguy (Matrox)	Ronald Anderson (Christie Digital)
Ben Cope (Intel)	Jed Deame (Nextera)	Ryan Wallenberg (Cobalt)
Brad Gilmer (VSF)	John Dale (Media Links)	Sara Seidel (Riedel)
Christian Toutant (Matrox)	Lynn Rowe (One World Tech)	Steve Kolta (Christie Digital)
Dan Holland (DHC)	Marc Levy (Macnica)	Thomas True (NVIDIA)
Daniel Bouquet (ANALOG WAY)	Marc-Antoine Massicotte (Matrox)	Tim Bruylants (intoPIX)
Danny Pierini (Matrox)	Mike Boucke (AJA)	Wes Simpson (LearnIPvideo)

1.2 About the Video Services Forum

The Video Services Forum, Inc. (www.videoservicesforum.org) is an international association dedicated to video transport technologies, interoperability, quality metrics and education. The VSF is composed of service providers, users and manufacturers. The organization's activities include:

- providing forums to identify issues involving the development, engineering, installation, testing and maintenance of audio and video services;
- exchanging non-proprietary information to promote the development of video transport service technology and to foster resolution of issues common to the video services industry;
- identification of video services applications and educational services utilizing video transport services;
- promoting interoperability and encouraging technical standards for national and international standards bodies.

The VSF is an association incorporated under the Not For Profit Corporation Law of the State of New York. Membership is open to businesses, public sector organizations and individuals worldwide. For more information on the Video Services Forum or this document, please call +1 929-279-1995 or e-mail opsmgr@videoservicesforum.org.

2 Conformance Notation

Normative text is text that describes elements of the design that are indispensable or contains the conformance language keywords: "shall", "should", or "may". Informative text is text that is potentially helpful to the user, but not indispensable, and can be removed, changed, or added

editorially without affecting interoperability. Informative text does not contain any conformance keywords.

All text in this document is, by default, normative, except the Introduction and any section explicitly labeled as "Informative" or individual paragraphs that start with "Note:"

The keywords "shall" and "shall not" indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.

The keywords "should" and "should not" indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keywords "may" and "need not" indicate courses of action permissible within the limits of the document.

The keyword “reserved” indicates a provision that is not defined at this time, shall not be used, and may be defined in the future. The keyword “forbidden” indicates “reserved” and in addition indicates that the provision will never be defined in the future.

A conformant implementation according to this document is one that includes all mandatory provisions ("shall") and, if implemented, all recommended provisions ("should") as described. A conformant implementation need not implement optional provisions ("may") and need not implement them as described.

Unless otherwise specified, the order of precedence of the types of normative information in this document shall be as follows: Normative prose shall be the authoritative definition; Tables shall be next; followed by formal languages; then figures; and then any other language forms.

3 Normative References

RFC 9293: Transmission Control Protocol (TCP)
TR-10-13:2024 IPMX Privacy Encryption Protocol (PEP)
RFC 8866: SDP: Session Description Protocol
RFC 4145: TCP-Based Media Transport in the Session Description Protocol (SDP)
RFC 3629: UTF-8, a transformation format of ISO 10646
AMWA IS-04 v1.3.2: "AMWA IS-04 NMOS Discovery and Registration Specification"
AMWA IS-05 v1.1.2: “AMWA IS-05 NMOS Device Connection Management Specification”
USB 2.0: Universal Serial Bus Specification Revision 2.0
IEEE SA, Registration Authority, CID (https://standards.ieee.org/products-programs/regauth/cid/)

4 Acronyms

CID	Company ID allocated by the IEEE registration authority
NMOS	Networked Media Open Specifications
SN	Receiver or Sender Serial Number.
USB	Universal serial bus.
WoL	Wake-on-LAN

5 Definitions

For the purposes of this document, the terms and definitions of VSF TR-10-1 and USB 2.0 specification apply. The following additional definitions also apply to this Technical Recommendation.

Bulk Endpoint	Endpoint that supports bulk transfers ("bmAttributes" bits 1:0 set to 0b10).
Control Endpoint	Endpoint that supports control transfers ("bmAttributes" bits 1:0 set to 0b00).
Endpoint	USB 2.0 Endpoint as defined in section 9.6.6 of the USB 2.0 specification.
Host	Processing device that is connected to a USB device such as mouse or keyboard through a IPMX USB-over-IP connection instead of being directly connected to the USB device.
Interrupt Endpoint	USB 2.0 Endpoint that supports interrupt transfers ("bmAttributes" bits 1:0 set to 0b11).
Isochronous Endpoint	USB 2.0 Endpoint that supports isochronous data transfers ("bmAttributes" bits 1:0 set to 0b01).
Receiver	Side that is connected to the Host of the IPMX USB stream.
Receiver Connection Data	This is the data used by a Sender to establish a TCP communication channel with a Receiver. This data is composed of the Receiver IP address of the First Control Channel with the Port address extracted from the Sender Connection Status.

Sender	Side that is connected to the physical USB device(s) of the IPMX USB stream.
Sender Connection Data	This is the data used by a Receiver to establish a TCP communication channel with a Sender. This data is contained in the SDP file.
TCP Accept	Refer to a Socket interface TCP accept operation.
TCP Connect	Refer to a Socket interface TCP connect operation.
UTF8	Unicode Transformation Format – 8-bit: Variable-length character encoding standard used for electronic communication as defined in RFC-3629.

6 Overview (Informative)

Typically, in an operating system, a device driver is used for communication between the application and the USB Bus Driver. The USB Bus Driver provides the interface between the Device Driver and the physical USB interface. This is shown in Figure 2 where the USB connection is in blue.

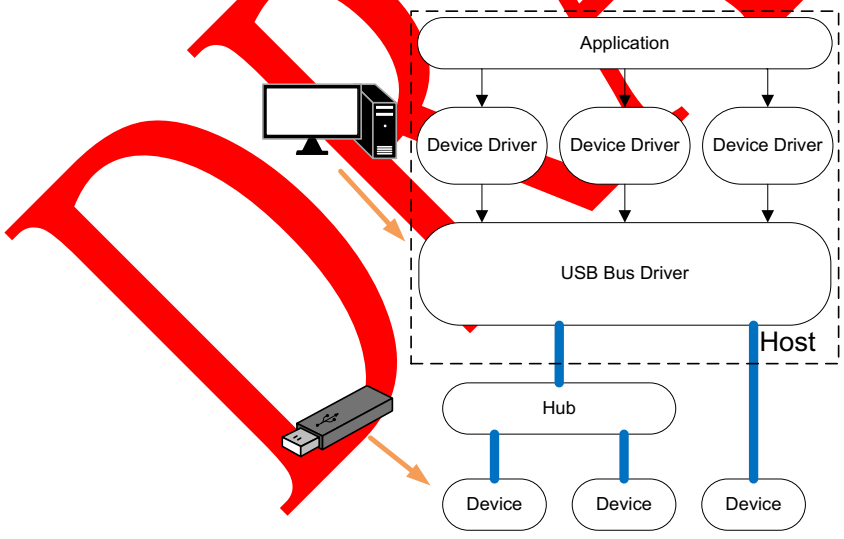

Figure 2: Traditional USB implementation

Figure 3 through Figure 6 show possible implementations for IPMX USB. The IPMX USB to IP gateway converts Host USB traffic into network traffic and is called the Receiver. The IPMX IP to USB gateway converts network traffic into Device USB traffic and is called the Sender. The IPMX USB to IP gateway is connected through the network to the IPMX IP to USB gateway.

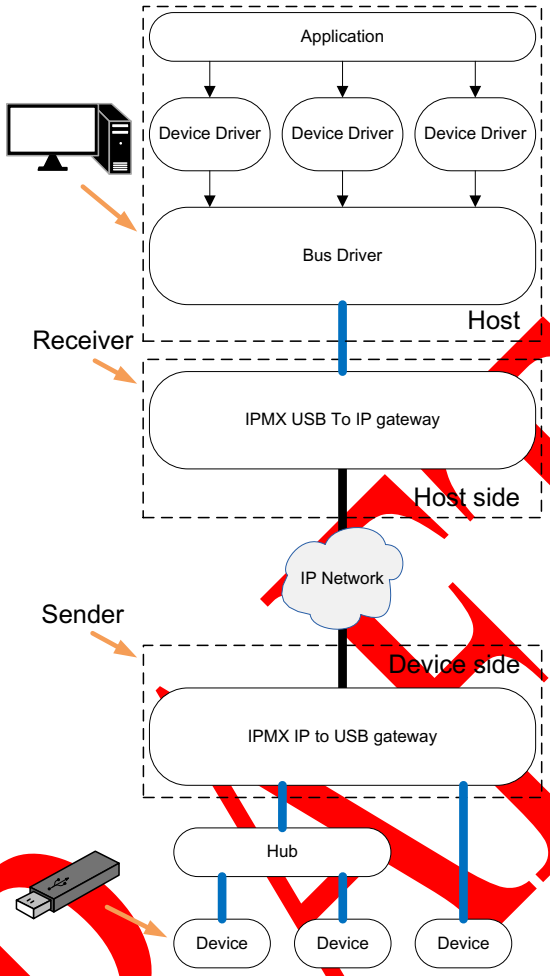

Figure 3: IPMX USB to IP gateway to IPMX IP to USB gateway

Figure 3 shows the case where a IPMX USB to IP gateway is connected to the Host USB port. The IPMX IP to USB gateway is connected to the USB devices. Both gateways are connected with a standard IP connection.

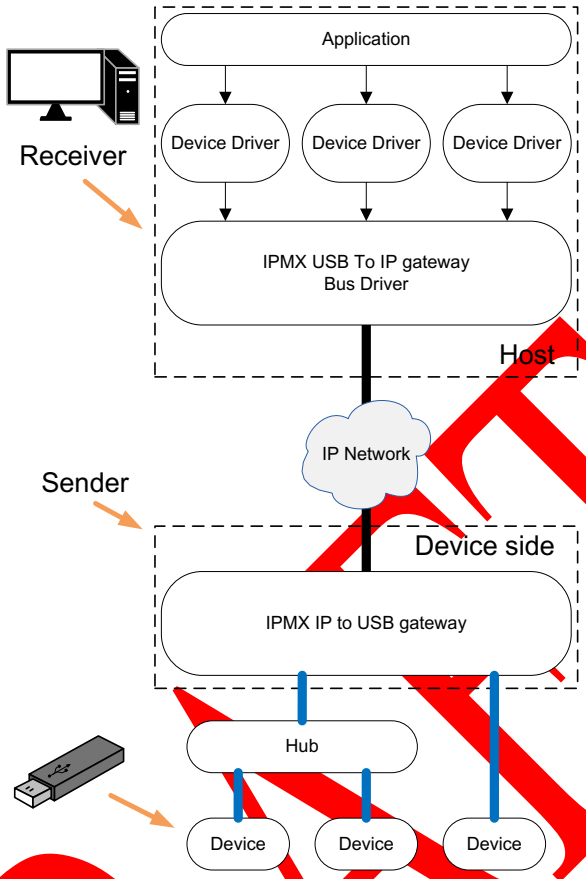

Figure 4: Integrated IPMX USB Host to IPMX IP to USB gateway

Figure 4 shows an alternative implementation where the IPMX USB to IP gateway is integrated into the Host. In such a case, the USB Bus Driver of the Host integrates the IPMX USB to IP gateway Bus Driver.

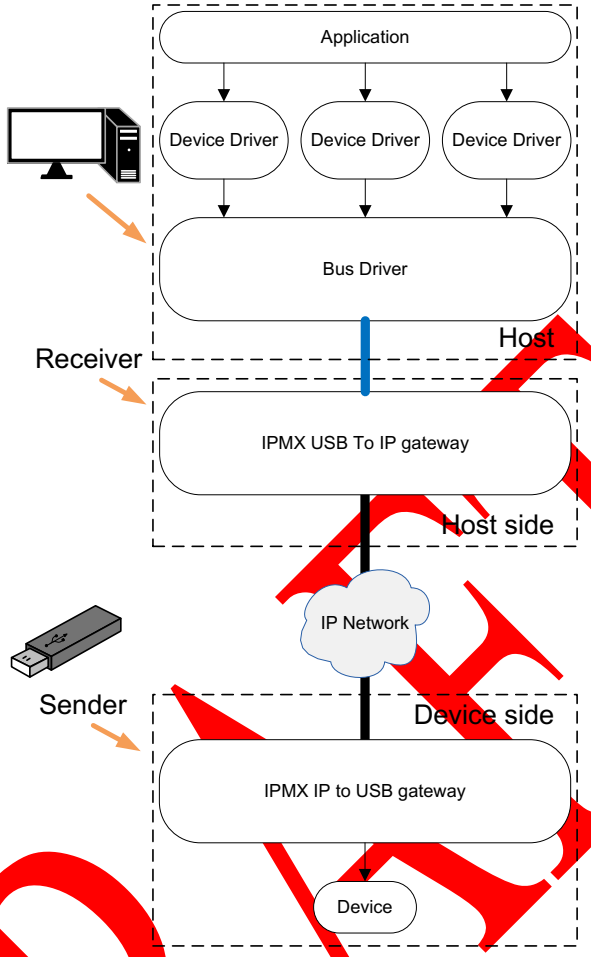

Figure 5: IPMX USB to IP gateway to Integrated IPMX USB device

Figure 5 shows an alternative implementation where the IPMX IP to USB gateway is integrated into the device. In such a case the device is a IPMX USB device instead of being a USB device.

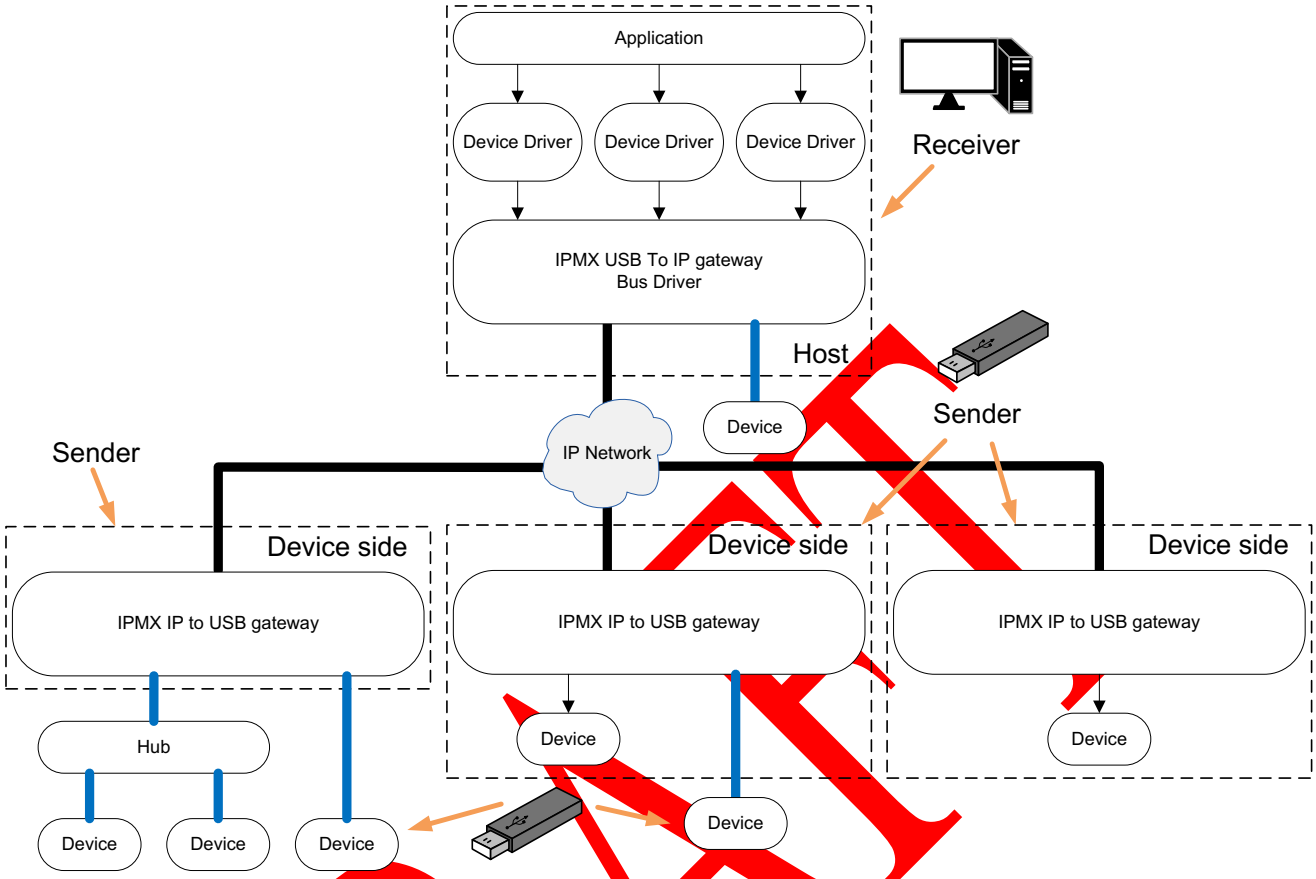

Figure 6: Complex example

There may be multiple variations on how the IPMX USB can be used. The common link is the IP interface between a Receiver and a Sender. In a last example, Figure 6 shows a more complex example. This example introduces three things:

1. The Host can be connected to IPMX USB devices and USB devices. This means that the Host USB driver includes a IPMX USB to IP gateway as well as interface to standard USB devices.
2. A Sender can integrate an IPMX USB device and an IPMX IP to USB gateway to interface with USB devices.
3. A Host can communicate to multiple IPMX USB devices from the same network interface.

IPMX USB allows the Host to access the full functionality of USB remote devices as if the USB device was directly attached to the Host. The IPMX USB to IP gateway or the IPMX USB to IP gateway Bus Driver and the IPMX IP to USB gateway provide an interface to remote devices by encapsulating peripheral bus request commands in IP packets and transmitting them across the network.

7 USB Hub compatibility issues (Informative)

The IPMX USB mostly operates as a USB Hub as defined by the USB 2.0 specification. However, there are some aspects of a Hub that the IPMX USB specification doesn't follow.

7.1 Latency

The USB specification clearly defines the maximum delay between a packet on the Upstream port of a Hub and the same packet on the downstream Hub of the port. Since the data on IPMX USB transit through an IP connection this maximum delay likely cannot be respected. Also, IPMX USB intends to support software implementation that also incurs extra delay.

Increasing the delay on packet may affect the response time. This may impact the performance of a device by reducing the effective bandwidth. An implementation may implement mechanisms to improve performance. It is out of the scope of this specification to describe and define such mechanisms.

7.2 Frame synchronization

According to the USB specification, a Hub replicates the frame timing of the Upstream port to all its downstream ports. However, an IPMX USB implementation does not include a mechanism to accomplish this synchronization. This lack of synchronization can cause issues, particularly with isochronous transfers, as they rely on receiving data at regular intervals based on the frame clock. It should be noted that this specification does not address how to resolve this problem.

Workarounds are possible. It is up to the implementor to pick the appropriate solution if Isochronous transfers are to be supported.

8 Communication Channels

Senders and Receivers exchange information through Communication Channels. Those communication channels are point to point. In a configuration where a Receiver is connected to multiple Senders, each Receiver to Sender connection uses its own set of communication channels.

Each connection between a Receiver and a Sender uses multiple communication channels: there shall be one control channel and there may be multiple data channels.

Communications channels shall use TCP protocol as defined by RFC-9293. The TCP data payload in Communications Channels shall be in the form of messages as defined in section 9 of this document.

8.1 Control Channel

The Control Channel shall be used for Receiver-to-Sender messages. This channel is setup using the information from the Sender Connection Data is included in the SDP file that shall be communicated to the Receiver by NMOS.

The Control Channel shall be used for the following messages from the Sender:

- Sender Connection Information
- Heartbeat
- Vendor Specific Information
- Vendor Specific Query

The Control Channel shall be used for the following messages from the Receiver:

- Sender Connection Status
- Vendor Specific Query Return

8.2 Data Channels

All the other communication channels are Data Channels. A Data Channel shall be created for each USB device. Data Channels shall be used for the USB data transfer of the USB device associated with the channel.

The Data Channels shall be used for the following messages from the Sender:

- USB Stream Information
- USB Stream Reset Return
- USB Submit Return (Control, Bulk, Interrupt, Isochronous)

The Data Channels shall be used for the following messages from the Receiver:

- USB Stream Status
- USB Stream Reset
- USB Submit (Control, Bulk, Interrupt, Isochronous)

8.3 Establishing a communication channel

The Control Channel shall be established with TCP using the Sender Connection Data. Data channels shall be established with TCP using the Receiver Connection Data.

Figure 7 shows how the communication channels are established.

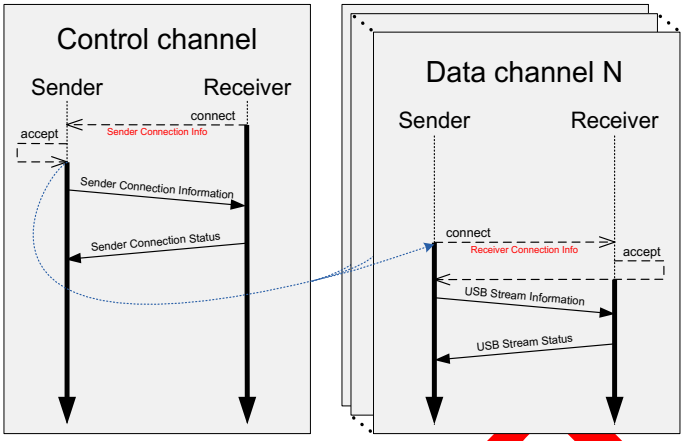

Figure 7: Communication channels

8.3.1.1 Sender-to-Receiver connection

Once a Receiver has received the Sender Connection Data from NMOS, it can connect to the Sender and wait for a command. After a successful TCP Connect, on reception of the Sender Connection Information message, the Receiver shall send a Sender Connection Status message. The Receiver is then ready to receive other messages.

The Sender accepts incoming TCP connection. After a successful TCP Accept, the Sender shall send a Sender Connection Information message and wait for a response. The Sender may send other messages once the Sender Connection Status message is received.

8.3.1.2 Data Channels connection

Once a Sender has accepted the incoming TCP connection on the Control channel, it may open Data Channels connections with the Receiver. The Sender shall open one Data Channel for every USB port connected to its downstream USB interface when a USB device is connected to this USB port. When the USB device is disconnected, the Data Channel connection shall be closed by the Sender.

For each Data Channel connection, the Sender shall connect to the Receiver. After a successful TCP Accept, the Sender shall send a USB Stream Information message and shall wait for the USB Stream Status message. Once the status is received, the Sender shall be ready to receive USB Submit messages.

For each Data Channel connection, the Receiver shall accept the TCP connection and then wait for a USB Stream Information message. The Receiver shall send a USB Stream Status message on reception of the USB Stream Information message. The Receiver shall not send USB Data Submit messages before the USB Stream Status message is sent.

9 IPMX USB TCP messages

IPMX USB TCP messages as shown in Figure 8 shall be stored in big endian format. The bit order of Figure 8 and all other figures representing messages, represent the transmission ordering where the MSB is transmitted first and the LSB is transmitted last. IPMX USB TCP messages should be encrypted and authenticated. Fields of the IPMX USB TCP message shall be as described in Table 1.

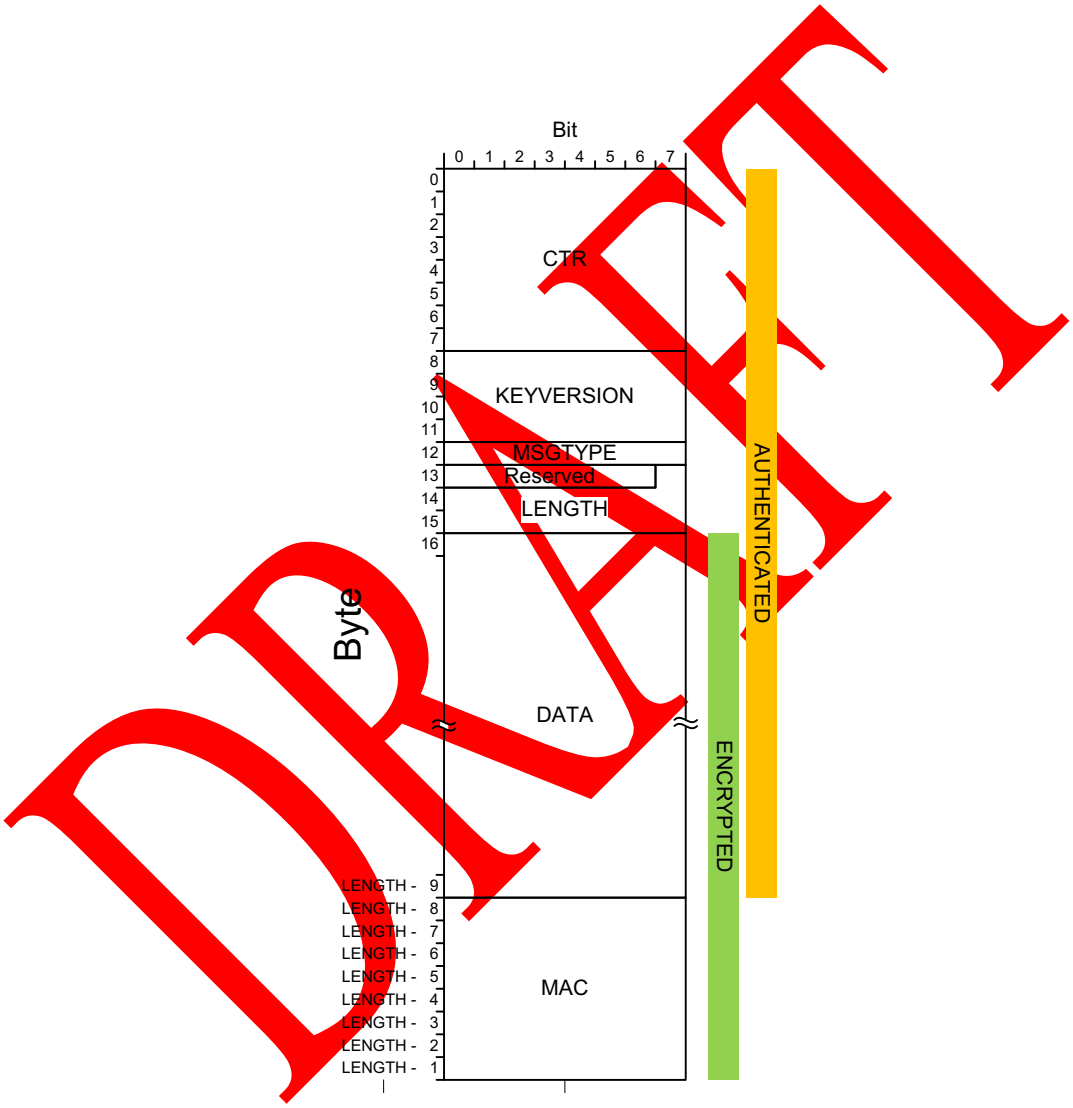

Figure 8: IPMX USB TCP messages

Field	Size	Description
CTR	64 bits	64-bit AES-CTR counter value. This field is authenticated. When encryption is disabled, this field shall be set to 0.
KEYVERSION	32 bits	32-bit Privacy cipher key version. This field corresponds to the key_version of the TR-10-13 Privacy Encryption Protocol. This field is authenticated. When encryption is disabled, this field shall be set to 0.
MSGTYPE	8-bit	8-bit message type. Bit 7 identify the channel type, Bit 0 the direction. and Bit 4 identify Submit – Submit Return. This field is authenticated. 0x00 Sender Connection Information 0x01 Sender Connection Status 0x02 Heartbeat 0x04 Vendor Specific Information 0x06 Vendor Specific Query 0x07 Vendor Specific Query Return 0x11 USB Wake-up Control 0x12 USB Enter Sleep 0x14 USB Resume Operation 0x80 USB Stream Information 0x81 USB Stream Status 0x82 USB Stream Reset Return 0x83 USB Stream Reset 0x90 USB Control Submit Return 0x91 USB Control Submit 0x92 USB Bulk Submit Return 0x93 USB Bulk Submit 0x94 USB Interrupt Submit Return 0x95 USB Interrupt Submit 0x96 USB Isochronous Submit Return 0x97 USB Isochronous Submit 0x98 USB Cancel Submit Return 0x99 USB Cancel Submit
Reserved	7 bits	7-bit. Those bits are reserved and shall be set to 0 before transmission and should be ignored on reception. This field is authenticated.
LENGTH	17 bits	17-bit length of the message in bytes. This field shall be set to a value between 24 and 131047. This field is authenticated.
DATA	(LENGTH – 24) bytes	IPMX USB data payload. The data payload is dependent on the DTYPE field and is defined for each value of DTYPE. This field is encrypted and authenticated.
MAC	64 bits	64 LSB of the CMAC of the plain-text packet without the MAC value itself. This field is encrypted. When encryption is disabled, this field shall be set to 0.

Table 1: IPMX USB TCP message fields

It is expected that most of the time encryption and authentication will be used. However, TR10-14 could be used without encryption and authentication. In this later case the same structure is used but this representation could be simplified as shown in Figure 9.

Figure 9:IPMX USB TCP message (without TR10-13)

9.1 Sender Connection Information

Sender Connection Information message shall be sent on the Control channel by a Sender when it has accepted a Receiver TCP connection. The DATA field of the message shall have a fixed size of 66 bytes as shown in Figure 10. Fields of the DATA field shall be as described in Table 2.

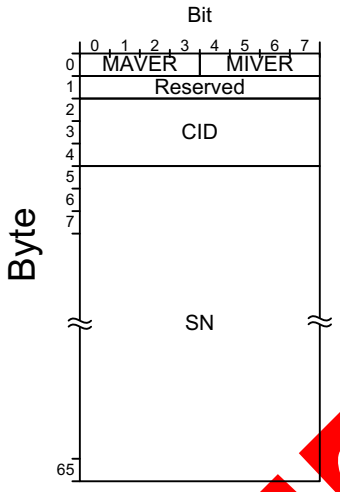

Figure 10: Sender Connection Information message

Field	Size	Description
MAVER	4 bits	Sender protocol Major Version supported. This value shall be 0 for this version of the specification.
MIVER	4 bits	Sender protocol Minor Version supported. This value shall be 0 for this version of the specification.
Reserved	8 bits	Those bits are reserved and shall be set to 0 by the Sender and shall be ignored by the Receiver.
CID	3 bytes	Sender Company ID.
SN	61 bytes	Sender Serial number. A UTF8 bytes string from left to right that identifies the Sender. All unused bytes at the end of the string shall be set to 0. This string shall be unique for this Sender Company ID.

Table 2: Sender Connection Information message DATA fields

9.2 Sender Connection Status

Sender Connection Status message shall be sent by a Receiver on reception of a Sender Connection Information message, only when the Receiver is connected to the Sender. The DATA field of the message shall have a fixed size of 68 bytes as shown in Figure 11. Fields of the DATA field shall be as described in Table 3.

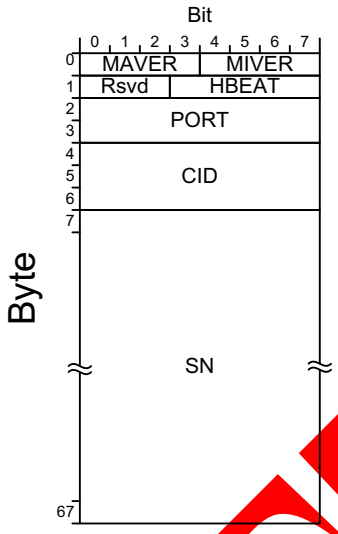

Figure 11: Sender Connection Status message

Field	Size	Description
MAVER	4 bits	Protocol Major Version. The combination of Protocol Major Version and Protocol Minor Version shall be smaller or equal to the combination of Protocol Major Version and Protocol Minor Version reported by the Sender. This value shall be 0 for this version of the specification.
MIVER	4 bits	Protocol Minor Version. The combination of Protocol Major Version and Protocol Minor Version shall be smaller or equal to the combination of Protocol Major Version and Protocol Minor Version reported by the Sender. This value shall be 0 for this version of the specification.
Rsvd	3 bits	Those bits are reserved and shall be set to 0 by a Receiver and shall be ignored by a Sender.
HBEAT	5 bits	Heartbeat rate index. The minimum valid value is 5 and the maximum valid value is 30. Conversion from index to second is explained in section 11 ”Heartbeat.”
PORT	16 bits	Receiver Port number that shall be used by a Sender to connect the Data channels.
CID	3 bytes	Receiver Company ID.
SN	61 bytes	Receiver Serial number. A UTF8 bytes string from left to right that identifies the Receiver. All unused bytes at the end of the string shall be set to 0. This string shall be unique for this Receiver Company ID.

Table 3: Sender Connection Status message DATA fields

9.3 Heartbeat

Heartbeat message shall be sent regularly by the Sender, as defined in section 11, to maintain the connection. There is no DATA field in the message.

9.4 Vendor Specific Information

Vendor Specific Information as shown in Figure 12 may be sent on the Control channel by a Sender to dispatch extra information that is not essential but that is complementary. A Receiver

shall accept all Vendor Specific Information messages. A Receiver that does not understand a Vendor Specific Information message shall ignore that message. Fields of the DATA field shall be as described in Table 4.

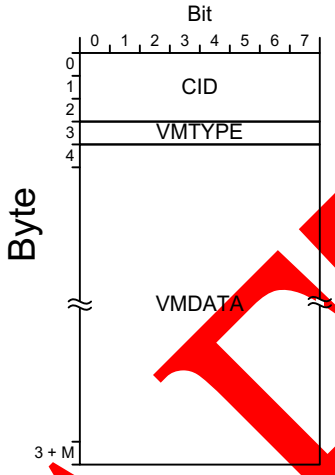

Figure 12: Vendor Specific message

Every Sender and Receiver shall support Vendor Specific Information message type 0. In this case the VMDATA is an informational UTF8 bytes string from left to right. The string shall be between 0 and 256 bytes.

Field	Size	Description
CID	3 bytes	Message Company ID.
VMTYPE	1 byte	Vendor Specific message type. 0 String information 1 - 15 Reserved for future use 16 - 255 Vendor specific message type. Refer to the vendor for the definition of VMDATA.
VMDATA	M bytes	Vendor Specific message data as defined by VMTYPE. M is the size and shall use all remaining bytes of the message

Table 4: Vendor Specific message DATA fields

9.5 Vendor Specific Query

Vendor Specific Query is sent on the Control channel by a Sender to request Vendor specific information. A Receiver shall accept all Vendor Specific Query messages and respond to the message. The DATA field of the message shall have a fixed size of 4 bytes as shown in Figure 13. Fields of the DATA field shall be as described in Table 5.

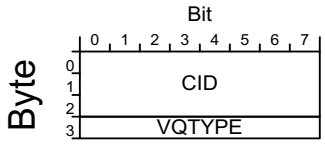

Figure 13: Vendor Specific Query message

The only VQTYPE defined for all Vendors is 0. In this case the VQDATA is an informational UTF8 bytes string from left to right. The content of the string is up to the Receiver but shall not exceed 256 bytes.

Field	Size	Description
CID	3 bytes	Message Company ID.
VQTYPE	1 byte	Vendor query type. 0 String information 1 - 15 Reserved for future use. 16 - 255 Vendor specific query type. Refer to the Vendor for the definition of VQDATA.

Table 5: Vendor Specific Query message DATA fields

9.6 Vendor Specific Query Return

Vendor Specific Query Return message as shown in Figure 14 is sent by a Receiver on reception of a Vendor Specific Query. Fields of the DATA field shall be as described in Table 6.

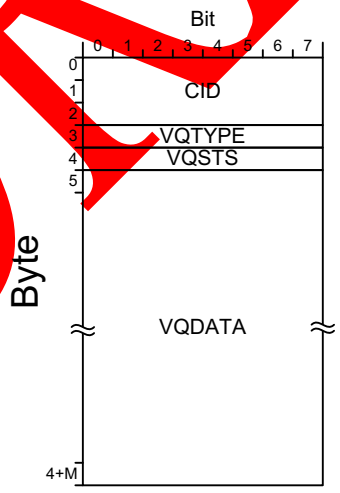

Figure 14: Vendor Specific Query Return message

A Receiver shall send this message for every Vendor Specific Query message received. A Vendor Specific Query Return message with VQSTS set to 255 and no VQDATA (M = 0) shall be returned for an unknown Vendor Specific Query message received by a Receiver.

All Receivers shall respond with no 'Error' to a Vendor Specific Query message with VQTYPE of 0. The content of the string is up to the Receiver but shall not exceed 256 bytes.

Field	Size	Description
CID	3 bytes	Message Data Company ID. For Query type 15 to 255, this value shall be identical to the CID of the equivalent Vendor Specific Query.
VQTYPE	1 byte	Vendor Specific Query type. 0 String information 1 - 15 Reserved for future use 16 - 255 Vendor VQTYPE. Refer to the Vendor for the definition of VQDATA.
VQSTS	1 byte	Vendor Specific Query Status. 0 OK 1 - 15 Reserved for future use 16 - 254 Vendor defined status. 255 Unknown Query
VQDATA	M bytes	Vendor Specific data as defined by VQTYPE. M is the size and shall use all remaining bytes of the message. If M is 0, there is no data to return.

Table 6: Vendor Specific Query Return message DATA fields

9.7 USB Stream Information

USB Stream Information message shall be sent by a Sender to create a USB stream after the Receiver has accepted a Receiver connection on a Data channel. The DATA field of the message shall have a fixed size of 66 bytes as shown in Figure 15. Fields of the DATA field shall be as described in Table 7.

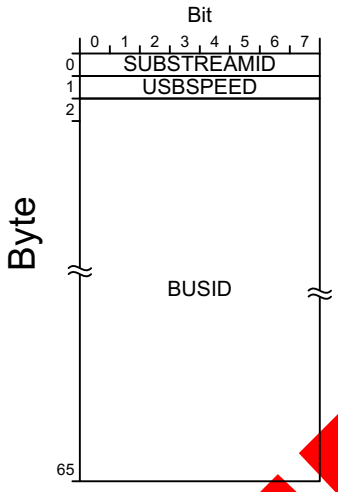

Figure 15: USB Stream Information message

Field	Size	Description
SUBSTREAMID	1 byte	Sub-stream ID. TCP channel Identifier. Bit 0 is used to identify the data direction where 0 is for Sender-to-Receiver transfer and 1 is for Receiver-to-Sender transfer. Bits 7 to 1 are used to identify the channel. Control channel has those bits set to 0. Values 1 to 127 are used for the data channels and are assigned by the Sender.
USBSPEED	1 byte	USB Speed. 0 Unknown 1 Low Speed 2 Full Speed 3 High Speed
BUSID	64 bytes	Bus ID. The Bus ID is a 64 UTF8 bytes string from left to right that identifies the USB device. All unused bytes at the end of the string shall be set to 0. An IPMX USB bus shall be uniquely identified by the combination of Sender's CID, Sender's SN, and Bus ID.

Table 7: USB Stream Information message DATA fields

9.8 USB Stream Status

USB Stream Status message shall be sent by a Receiver when it has received a USB Stream Information message. The DATA field of the message shall have a fixed size of 1 byte as shown in Figure 16. Field of the DATA field shall be as described in Table 8.

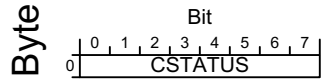

Figure 16: USB Stream Status message

Field	Size	Description
CSTATUS	1 byte	Connection status. A CSTATUS of 0 indicates that there is no error (OK status). A CSTATUS of 255 indicates an error. Other values are reserved.

Table 8: USB Stream Status message DATA fields

9.9 USB Stream Reset

USB Stream Reset message may be sent by the Receiver to reset the USB port. This message can be sent at any time after the Receiver has received a USB Stream Information. There is no data in the message. After sending a USB Stream Reset message, the Receiver shall wait for a USB Stream Reset Return message before sending any message to that Port.

Refer to section 7.1.7.5 “Reset Signaling” of the USB 2.0 Specification for more information about the reset.

9.10 USB Stream Reset Return

USB Stream Reset Return message shall be sent by the Sender to confirm that the USB Stream Reset has been processed. There is no data in the message.

9.11 USB Wake-up Control

USB Stream Wake-up Control message may be sent by a Receiver to control the Wake-on-LAN state of the Sender. The default Wake-on-LAN state is ‘Disable’. This message can be sent at any time after the Receiver has received a Sender Connection Information message. The DATA field of the message is shown in Figure 17 and shall have a size between 1 and 7 bytes. Fields of the DATA field shall be as described in Table 9.

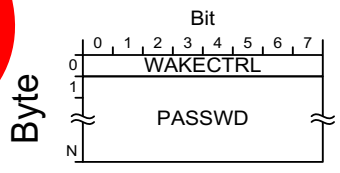

Figure 17: USB Wake-up Control message

Field	Size	Description
WAKECTRL	1 byte	Control if the WoL is enable: 0 Disable. Magic Packet shall not be sent on USB Wake-up. 1 Enable. Magic Packet is sent on USB Wake-up.
PASSWD	N byte	Magic packet password. The value of N shall be between 0 byte (no password) to 6 bytes. This password is added at the end of the magic packet.

Table 9: USB Wake-up Control message DATA fields

9.12 USB Enter Sleep

USB Enter Sleep message shall be sent by the Sender when all the USB ports connected to the Sender enter sleep mode.

Once the message is sent, when USB Wake-up Control enables WoL, the Sender shall close all connections to the Receiver. Once the message is received, the Receiver shall close all connections to the Sender and should not try to reconnect to the Sender until a WoL is received. There is no DATA field in the message.

9.13 USB Resume operation

USB Resume operation message shall be sent by the Sender when a USB Enter Sleep message is the last message sent by the Sender, the USB Wake-up Control disable WoL, and one of the USB ports connected to the Sender resumes signaling.

9.14 USB Control Submit

USB Control Submit message can be sent by a Receiver when it has sent a USB Stream Status message. USB Control Submit message shall be sent only by Control Endpoints. The DATA field of the message is shown in Figure 18. Fields of the DATA field shall be as described in Table 10.

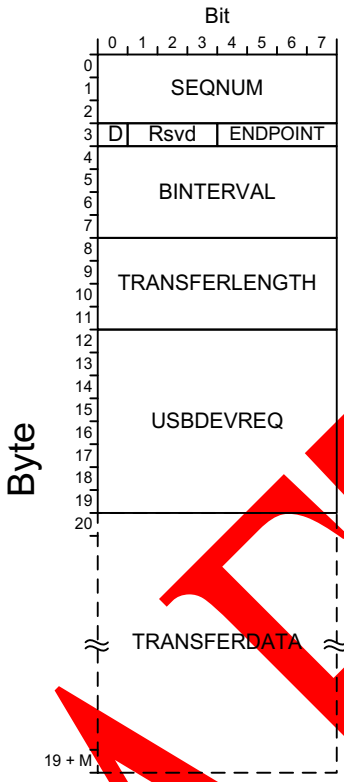

Figure 18: USB Control Submit message

Field	Size	Description
SEQNUM	3 bytes	Sequential number that identifies the USB Submit. The value for the first request sent on this data channel shall be 0. This value shall be incremented by one from the SEQNUM value of the previous Submit message sent on this data channel.
ENDPOINT	4 bits	Device Endpoint number.
Rsvd	3 bits	Reserved. This field shall be set to 0 by a Receiver and shall be ignored by a Sender.
D	1 bit	Direction. This bit shall be set to 0 to indicate the OUT (Receiver-to-Sender) Endpoint and shall be set to 1 to indicate the IN Endpoint (Sender-to-Receiver).
BINTERVAL	4 bytes	Interval for polling Endpoint for data transfers. Refer to Table 9-13 of USB 2.0 specification for more information about this field.
TRANSFERLENGTH	4 bytes	Specifies the length of the data buffer. This field shall be set to the value of 'wLength' as given by Table 9-2 of the USB 2.0 specification. This field has a different meaning depending on the direction of data transfer as indicated by the D field. On Receiver-to-Sender Transfer Data (D = 0), this field shall be set to the exact amount of TRANFERDATA. On Sender-to-Receiver data transferred (D = 1), this field shall indicate the maximum size of the returned data to prevent overflow of the Receiver buffer.
USBDEVREQ	8 bytes	USB device request as defined in section 9.3 of the USB 2.0 specification.
TRANFERDATA	M bytes	When D is 0, this is an array of M bytes of data where M is TRANSFERLENGTH. When D is 1, there is no TRANSFERDATA and M is 0. The content of the TRANFERDATA depends on the 'bmRequestType' and 'bRequest' fields of USBDEVREQ and is defined in Table 9.3 of the USB 2.0 specification.

Table 10: USB Control Submit message DATA fields

9.15 USB Bulk Submit

USB Bulk Submit message can be sent by a Receiver when it has sent a USB Stream Status message. USB Bulk Submit message shall be sent only on Bulk Endpoint. The DATA field of the message is shown in Figure 19. Fields of the DATA field shall be as described in Table 11.

For Receiver-to-Sender data transfer, the TRANFERDATA shall be the Bulk Data as defined in section 4.7.2 “Bulk Transfers” of the USB 2.0 specification.

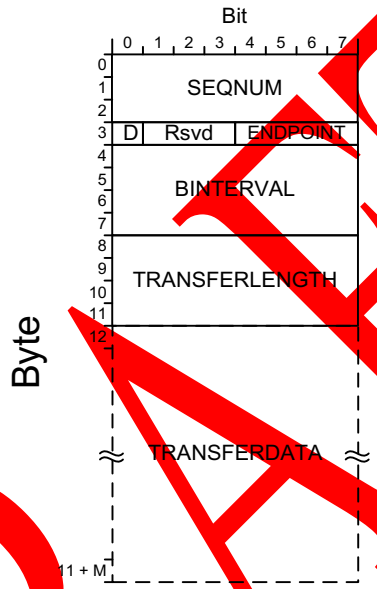

Figure 19: USB Bulk or Interrupt Submit message

Field	Size	Description
SEQNUM	3 bytes	Sequential number that identifies the USB Submit. The value for the first request sent on this data channel shall be 0. This value shall be incremented by one from the SEQNUM value of the previous Submit message sent on this data channel.
ENDPOINT	4 bits	Device Endpoint number.
Rsvd	3 bits	Reserved. This field shall be set to 0 by a Receiver and shall be ignored by a Sender.
D	1 bit	Direction. This bit shall be set to 0 to indicate the OUT (Receiver-to-Sender) Endpoint and shall be set to 1 to indicate the IN Endpoint (Sender-to-Receiver).
BINTERVAL	4 bytes	Interval for polling Endpoint for data transfers. Refer to Table 9-13 of USB 2.0 specification for more information about this field.
TRANSFERLENGTH	4 bytes	Specifies the length of the data buffer. This field has a different meaning depending on the direction of data transfer as indicated by the D field. On Receiver-to-Sender Transfer Data (D = 0), This field shall be set to the exact amount of TRANFERDATA. On Sender-to-Receiver data transferred (D = 1), this field shall indicate the maximum size of the returned data to prevent overflow of the Receiver buffer.
TRANFERDATA	M bytes	When D is 0, this is an array of M bytes of data where M is TRANSFERLENGTH. When D is 1, there is no TRANSFERDATA and M is 0.

Table 11: USB Bulk or Interrupt Submit message DATA fields

9.16 USB Interrupt Submit

USB Interrupt Submit message can be sent by a Receiver when it has sent a USB Stream Status message. USB Interrupt Submit message shall be sent only on Interrupt Endpoint. The DATA field of the message is shown in Figure 19. Fields of the DATA field shall be as described in Table 11.

For Receiver-to-Sender data transfer, the TRANFERDATA shall be the Interrupt Data as defined in section 4.7.3 “Interrupt Transfers” of the USB 2.0 specification.

9.17 USB Isochronous Submit

USB Isochronous Submit message can be sent by a Receiver when it has sent a USB Stream Status message. USB Isochronous Submit message shall be sent only on Isochronous Endpoint. The DATA field of the message is shown in Figure 20. Fields of the DATA field shall be as described in Table 12.

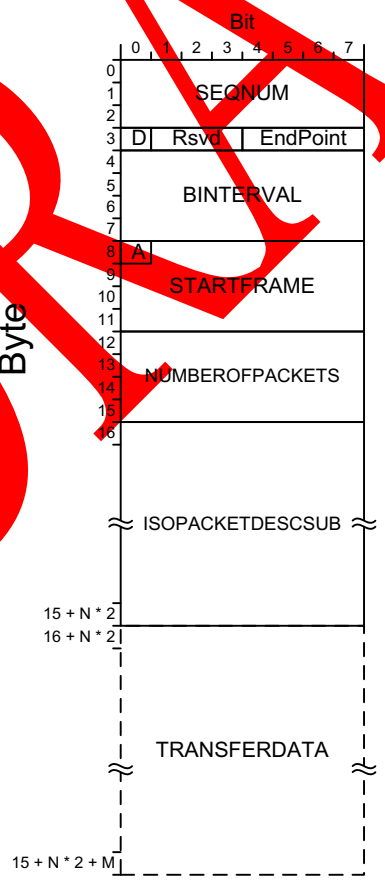

Figure 20: USB Isochronous Submit message

Field	Size	Description
SEQNUM	3 bytes	Sequential number that identifies the USB Submit. The value for the first request sent on this data channel shall be 0. This value shall be incremented by one from the SEQNUM value of the previous Submit message sent on this data channel.
ENDPOINT	4 bits	Device Endpoint number.
Rsvd	3 bits	Reserved. This field shall be set to 0 by a Receiver and shall be ignored by a Sender.
D	1 bit	Direction. This bit shall be set to 0 to indicate the OUT (Receiver-to-Sender) Endpoint and shall be set to 1 to indicate the IN Endpoint (Sender-to-Receiver).
BINTERVAL	4 bytes	Interval for polling Endpoint for data transfers. Refer to Table 9-13 of USB 2.0 specification for more information about this field.
A	1 bit	When this bit is 0 the Isochronous transfer shall be started at the frame specified by STARTFRAME. When this bit is 1, the Isochronous transfer shall be started as soon as possible.
STARTFRAME	31 bits	Specifies the frame number of the first packet of this submit shall begin on when A is 0. This variable must be within a system-defined range of the current frame. If A is 1, this field shall be set to 0 by the Receiver and shall not be used by the Sender.
NUMBEROFPACKETS	4 bytes	Number of isochronous frame of this Submit message.
ISOPACKETDESCSUB	N * 2 bytes	This is an array of N ISOLENGTH as defined in Figure 22 and Table 13 where N is NUMBEROFPACKETS.
TRANFERDATA	M bytes	When D is 0, this is an array of M bytes of data where M shall be the size of the isochronous TRANFERDATA of Figure 21. When D is 1, there is no TRANSFERDATA and M is 0.

Table 12: USB Isochronous Submit message DATA fields

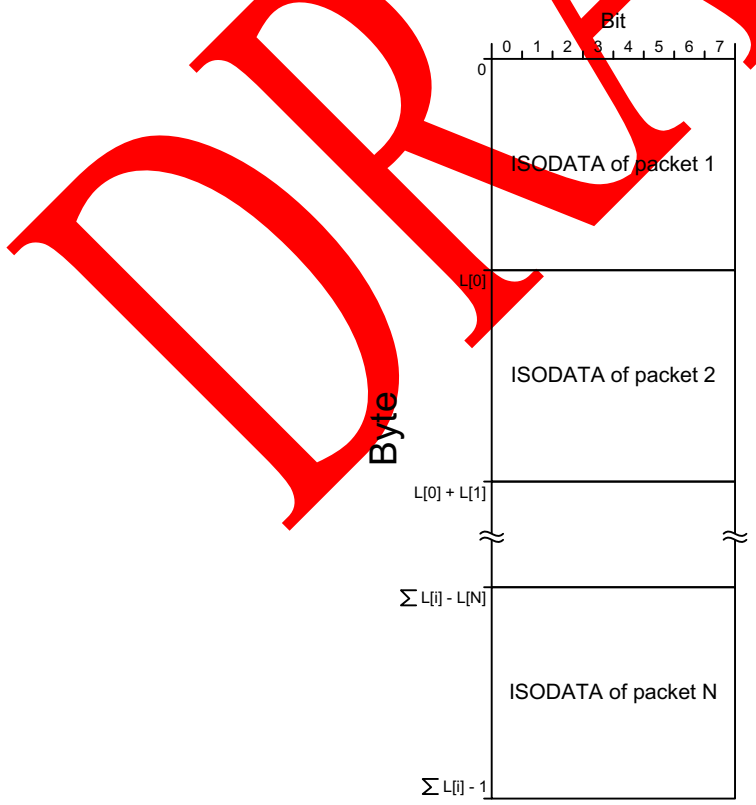

Figure 21: Isochronous TRANSFERDATA

For Receiver-to-Sender data transfer, the TRANFERDATA is the concatenation of the isochronous data for all isochronous packets of this Submit as shown in Figure 21. The message transports the information for multiple isochronous packets where [i] represents the index that selects an isochronous packet starting with 0 for the first packet and ending with N – 1 for the last packet. The size of each isochronous data L[i] is given by the corresponding field ISOLENGTH[i] as shown in Figure 22 and Table 13. The ISODATA is defined in section 4.7.4 "Isochronous Transfers" of the USB 2.0 specification.

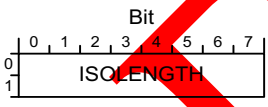

Figure 22: ISOLENGTH

Field	Size	Description
ISOLENGTH	16 bits	Isochronous Data size in bytes. The maximum value is 1024. On Receiver-to-Sender Transfer Data (D = 0), This field shall be set to the exact amount of data of the corresponding packet. On Sender-to-Receiver data transferred (D = 1), this field shall indicate the maximum size of the returned data in the corresponding packet to prevent overflow of the Receiver buffer.

Table 13: ISOLENGTH field

9.18 USB Control Submit Return

USB Control Submit Return message shall be sent by a Sender in response to a USB Control Submit message. The DATA field of the message is shown in Figure 23. Fields of the DATA field shall be as described in Table 14.

For Sender-to-Receiver data transfer, the TRANFERDATA is the data as given by Table 9-3 of the USB 2.0 specification.

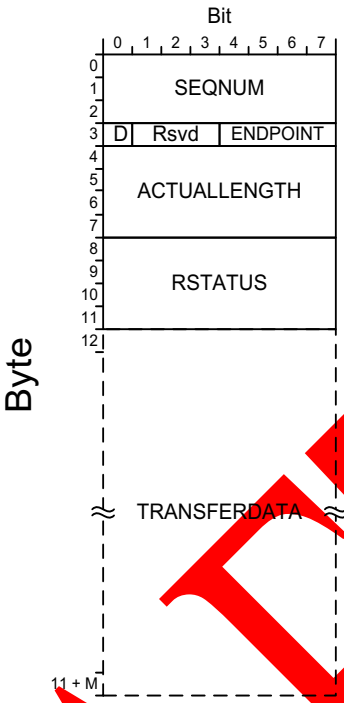

Figure 23: USB Control, Bulk, or Interrupt Submit Return message

Field	Size	Description
SEQNUM	3 bytes	Sequential number that identifies the USB Submit Return. This value shall match the SEQNUM value of the corresponding USB Submit message.
ENDPOINT	4 bits	Device Endpoint number. This value shall match the ENDPOINT value of the corresponding USB Submit message.
Rsvd	3 bits	Reserved. This field shall be set to 0 by a Sender and shall be ignored by a Receiver.
D	1 bit	Direction. This value shall match the D value of the corresponding USB Submit message.
ACTUALLENGTH	4 bytes	Specifies the length of the TRANFERDATA. When D is 0, this field shall be 0. When D is 1, the Sender shall never return more data than is indicated by the TRANSFERLENGTH value of the corresponding Submit request; it may return less. For USB Control Submit, the TRANSFERLENGTH field shall be set to the value of wLength as given by Table 9-3 of the USB 2.0 specification. If this field is zero, there is no data.
RSTATUS	4 bytes	Request Status. Refer to appendix A.1 for RSTATUS status code values.
TRANFERDATA	M bytes	This is an array of M bytes of data where M is ACTUALLENGTH. The content of the TRANFERDATA depends on the Submit Return type.

Table 14: USB Control, Bulk, or Interrupt Submit Return message DATA fields

9.19 USB Bulk Submit Return

USB Bulk Submit Return message shall be sent by a Sender in response to a USB Bulk Submit message. The DATA field of the message is shown in Figure 23. Fields of the DATA field shall be as described in Table 14.

For Sender-to-Receiver data transfer, the TRANFERDATA is the Bulk data as defined in section 4.7.2 “Bulk Transfers” of the USB 2.0 specification.

9.20 USB Interrupt Submit Return

USB Interrupt Submit Return message shall be sent by a Sender in response to a USB Interrupt Submit message. The DATA field of the message is shown in Figure 23. Fields of the DATA field shall be as described in Table 14.

For Sender-to-Receiver data transfer, the TRANFERDATA is the Interrupt data as defined in section 4.7.3 “Interrupt Transfers” of the USB 2.0 specification.

9.21 USB Isochronous Submit Return

USB Isochronous Submit Return message shall be sent by a Sender in response to a USB Isochronous Submit message. The DATA field of the message is shown in Figure 24. Fields of the DATA field shall be as described in Table 15.

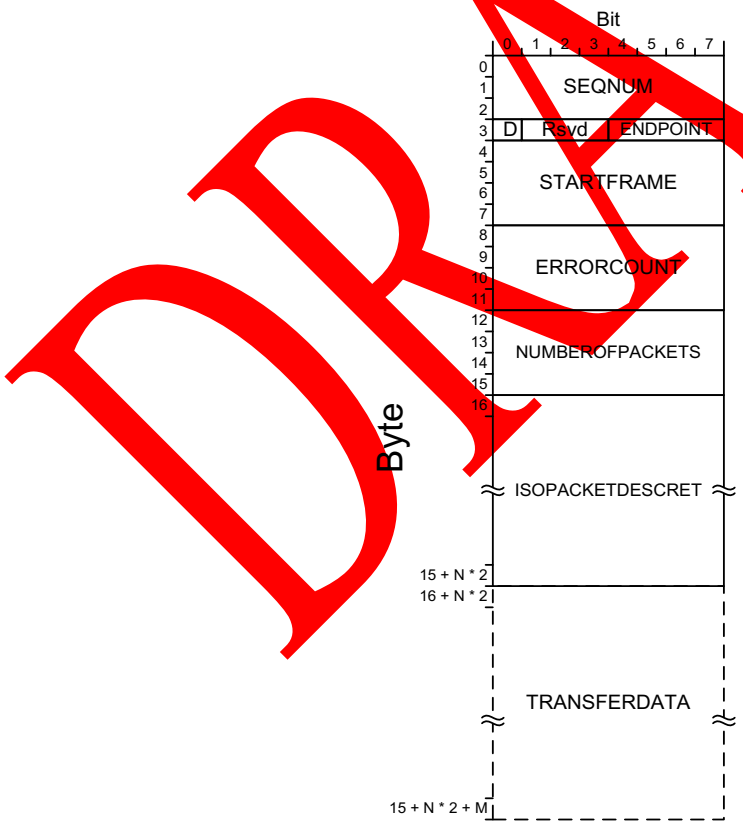

Figure 24: USB Isochronous Submit Return message

Field	Size	Description
SEQNUM	3 bytes	Sequential number that identifies the USB Submit Return. This value shall match the SEQNUM value of the corresponding USB Submit message.
ENDPOINT	4 bits	Device Endpoint number. This value shall match the ENDPOINT value of the corresponding USB Submit message.
Rsvd	3 bits	Reserved. This field shall be set to 0 by a Sender and shall be ignored by a Receiver.
D	1 bit	Direction. This value shall match the D value of the corresponding USB Submit message.
STARTFRAME	4 bytes	Specifies the frame number of the first packet of this submit return.
ERRORCOUNT	4 bytes	Number of packets of this Submit Return that completed with an error condition.
NUMBEROFPACKETS	4 bytes	Number of Isochronous frame of this Submit Return. This value shall be identical to the NUMBEROFPACKETS value of the corresponding USB submit message.
ISOPACKETDESCRET	N * 6 bytes	This is an array of N ISOPACKETRETSTRUC structure as defined in Figure 25 and Table 16 where N is NUMBEROFPACKETS.
TRANFERDATA	M bytes	When D is 1, this is an array of M bytes of data where M shall be the size of the Isochronous Transfer Data of Figure 21. When D is 0, there is no TRANSFERDATA and M is 0.

Table 15: USB Isochronous Submit Return message DATA fields

for Sender-to-Receiver data transfer, the TRANFERDATA is the concatenation of the Isochronous data of all Isochronous transfer of this Submit Return as shown in Figure 21. The message transports the information for multiple Isochronous packets where [i] represents the index that selects an Isochronous packet starting with 0 for the first packet and ending with N – 1 for the last packet. The size of each Isochronous Data L[i] is given by the ISOACTUALLENGTH value of the corresponding entry of ISOPACKETDESCRET as shown in Figure 25 and Table 16. The Isochronous Data is defined in section 4.7.4 “Isochronous Transfers” of the USB 2.0 specification.

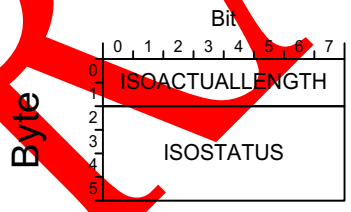

Figure 25: ISOPACKETRETSTRUC

Field	Size	Description
ISOACTUALLENGTH	2 bytes	Isochronous Data size in bytes. When D is 1, this value shall be smaller or equal to the value of ISOLENGTH[i] of the correspondent Submit. When D is 0, this value shall be 0.
ISOSTATUS	4 bytes	Isochronous status. Refer to appendix A.1 for ISOSTATUS status code values.

Table 16: ISOPACKETRETSTRUC fields

9.22 USB Cancel Submit

USB Cancel Submit message can be sent by a Receiver when it has sent a USB Submit message. The DATA field of the message is shown in Figure 26. Fields of DATA field shall be as described in Table 17.

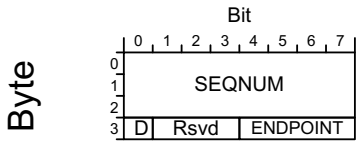

Figure 26: USB Cancel Submit message

Field	Size	Description
SEQNUM	3 bytes	Sequential number that identifies the USB Cancel Submit. This value shall match the SEQNUM of the USB Submit to Cancel.
ENDPOINT	4 bits	Device Endpoint number. This value shall match the ENDPOINT of the USB Submit to Cancel.
Rsvd	3 bits	Reserved. This field shall be set to 0 by a Sender and shall be ignored by a Receiver.
D	1 bit	Direction. This value shall match the D of the USB Submit to Cancel.

Table 17: USB Cancel Submit message DATA fields

9.23 USB Cancel Submit Return

USB Cancel Submit Return message shall be sent by a Sender in response to a USB Cancel Submit message. The DATA field of the message shall have a fixed size of 7 bytes as shown in Figure 27. Fields of the DATA field shall be as described in Table 18.

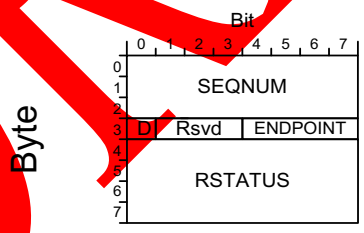

Figure 27: USB Cancel Submit Return message

Field	Size	Description
SEQNUM	3 bytes	Sequential number that identifies the USB Submit Return. This value shall match the SEQNUM value of the corresponding USB Cancel Submit message.
ENDPOINT	4 bits	Device Endpoint number. This value shall match the ENDPOINT value of the corresponding USB Cancel Submit message.
Rsvd	3 bits	Reserved. This field shall be set to 0 by a Sender and shall be ignored by a Receiver.
D	1 bit	Direction. This value shall match the D value of the corresponding USB Cancel Submit message.
RSTATUS	4 bytes	Request Status. Refer to appendix A.1 for RSTATUS status code values.

Table 18: USB Cancel Submit Return message DATA fields

10 Sharing access to USB devices

There are multiple cases: a Sender can be connected to a Receiver, Multiple Senders can be connected to a Receiver, or a Sender can be connected to multiple Receivers. In all cases, the communication between the Sender and the Receiver is unicast such that if multiple Senders are connected to a Receiver, or if a Sender is connected to multiple Receivers, each pair of Sender/Receiver shall use a different set of communication channels. A Receiver shall connect to each individual Sender. Reciprocally, a Sender should accept the connection of each individual Receiver.

Senders and Receivers may have a limit on the number of simultaneous connections they can support. When this limit is reached, a Receiver shall not try to connect to a new Sender and a Sender shall fail to accept a Receiver's connection.

11 Heartbeat

A Sender shall send a Heartbeat message once per Heartbeat_Period. The Heartbeat_Period is specified by HBEAT field of the Sender Connection Status message. The following formula shall be used to convert HBEAT index into seconds:

$$\text{Heartbeat\_Period} = 5 \text{ sec.} * 1.25^{\text{HBEAT}}$$

A Receiver shall consider that the Sender is not responding if a Heartbeat message is not received on the control channel within two times the Heartbeat_Period as calculated from the last Heartbeat message.

A non-responsive Sender shall be disconnected by the Receiver if it is not responding, and the Sender device shall loose its exclusive state.

12 Encryption

When encryption and authentication are not used, the privacy attribute shall not appear in the Sender's SDP transport file. The Sender's privacy parameters shall either not appear in the NMOS transport parameters or the protocol and mode of the NMOS transport parameters shall be set to NULL.

When encryption and authentication are used, TR-10-13 Privacy Encryption Protocol (PEP) shall be used for managing the encryption keys and protecting the USB control and data channels. The privacy attribute shall appear in the SDP transport file. The privacy parameters shall appear in the NMOS transport parameters and the protocol and mode of the NMOS transport parameters

shall not be set to NULL. Senders and Receivers shall support the **AES-128-CTR_CMAC-64-AAD** mode.

Other encryption and authentication methods compatible with the message layout may be used. The mode parameter shall be one of AES-128-CTR_CMAC-64-AAD, AES-256-CTR_CMAC-64-AAD, ECDH_AES-128-CTR_CMAC-64-AAD, ECDH_AES-256- CTR_CMAC-64-AAD.

The CMAC function used for authentication shall use the privacy cipher key. When the key is 128 bits the CMAC function shall use the AES-128 block cipher. When the key is 256 bits the CMAC function shall use the AES-256 block cipher.

Figure 8 shows which portion of the message is authenticated and which portion of the message is encrypted. The message is first signed and then encrypted storing the encrypted MAC as the last 8 bytes of the message.

During the encryption process, the portion of the message to be encrypted shall be divided in slices of 16 bytes and padding bytes are added at the end if the message is not a multiple of 16 bytes. These extra padding bytes shall be discarded and shall not be considered as part of the message.

The encryption key shall correspond to the privacy_key in the Privacy Key Derivation section of the PEP specification. Within channels, SUBSTREAMID is paired as specified by the PEP specification for bidirectional streams. SUBSTREAMID shall be unique at any time such that the iv' is unique as required by PEP.

The KEYVERSION and the CTR values are included in the header of the message as shown in Figure 8. At each TCP session, when the connection is accepted, the Sender shall increment the KEYVERSION and shall reset the CTR value to 0.

13 Wake-on-LAN

Wake-on-LAN shall be generated by the Sender when a USB resume condition is detected on the Sender USB connector and the Wake-on-LAN is activated by the Wake-up Control message. In such a case a Wake-on-LAN packet is sent from the Sender to the Receiver with the data included in the Wake-up Control message.

14 USB-SDP definition

The SDP shall follow RFC 4145 with the following restrictions:

The media space port of the media-field shall be set to “application/usb”

m=application <port> TCP usb

<port>: TCP server port of the Sender

When Encryption and Authentication are used, the privacy attribute as defined in TR10-13 shall be present with:

<protocol>: USB_KV

<mode>: Valid mode as defined in section 12

The ‘role’ of the ‘setup’ attribute shall be “passive”.

a=setup:passive

14.1 SDP example (informative)

v=0

o=- 1689604996 1246597654 IN IP4 192.168.152.182

s= Device MTX12345 usb 0

t=0 0

m=application 27502 TCP usb

c=IN IP4 192.168.152.182

a=privacy:protocol=USB_KV; mode=AES-128-CTR_CMAC-64-AAD; iv=181d3f3236be89b0; key_generator=836b4d6eb7cbd16055c6c827237faf97; key_version=2d9bc4e0; key_id=0001020304050607

a=setup:passive

15 IS-05 NMOS Transport Parameter

15.1 Sender

This section describes the USB Sender transport parameters. At a minimum, a USB Sender shall support “source_ip”, “source_port” in addition to the PEP privacy parameters when PEP is used.

"source_ip": IP address of the TCP server of the Sender.

"source_port": Port of the TCP server of the Sender.

15.2 Receiver

This section describes the USB Receiver transport parameters. A USB Receiver shall support at least “source_ip”, “interface_ip” and “source_port” in addition to the PEP privacy parameters when PEP is used.

"source_ip": IP address of the TCP server of the Sender.

"source_port": Port of the TCP server of the Sender.

"interface_ip": IP address the TCP client of the Receiver.

Appendix A Software interface information

A.1 Status code

The status code reports the status of the transfer as shown in Table 19. The status code shall be IPMX_USB_STATUS_SUCCESS if the transfer has completed successfully. When the transfer fail, Senders shall send an error code from Table 17. If an implementation has an error code that doesn’t match one of the entries in Table 17, IPMX_USB_STATUS_UNKNOWN_ERROR shall be used.

Value	Name	Description
0x00000000	IPMX_USB_STATUS_SUCCESS	Data transfer completed successfully.
0xC0000001	IPMX_USB_STATUS_CRC	CRC error (defined for backward compatibility with the USB 1.0).
0xC0000002	IPMX_USB_STATUS_BTSTUFF	BTS error (defined for backward compatibility with the USB 1.0).
0xC0000003	IPMX_USB_STATUS_DATA_TOGGLE_MISMATCH	Data toggle mismatch.
0xC0000004	IPMX_USB_STATUS_STALL_PID	The device returned a stall packet identifier (defined for backward compatibility with the USB 1.0).

0xC0000005	IPMX_USB_STATUS_DEV_NOT_RESPONDING	The device is not responding (defined for backward compatibility with the USB 1.0).
0xC0000006	IPMX_USB_STATUS_PID_CHECK_FAILURE	The device returned a packet identifier check failure (defined for backward compatibility with the USB 1.0).
0xC0000007	IPMX_USB_STATUS_UNEXPECTED_PID	The device returned an unexpected packet identifier error (defined for backward compatibility with the USB 1.0).
0xC0000008	IPMX_USB_STATUS_DATA_OVERRUN	The device returned a data overrun error (defined for backward compatibility with the USB 1.0).
0xC0000009	IPMX_USB_STATUS_DATA_UNDERRUN	The device returned a data underrun error (defined for backward compatibility with the USB 1.0).
0xC000000C	IPMX_USB_STATUS_BUFFER_OVERRUN	The device returned a buffer overrun error (defined for backward compatibility with the USB 1.0).
0xC000000D	IPMX_USB_STATUS_BUFFER_UNDERRUN	The device returned a buffer underrun error (defined for backward compatibility with the USB 1.0).
0xC000000F	IPMX_USB_STATUS_NOT_ACCESSED	The USB stack could not access the device (defined for backward compatibility with the USB 1.0).
0xC0000010	IPMX_USB_STATUS_FIFO	The device returned a FIFO error (defined for backward compatibility with the USB 1.0).
0xC0000011	IPMX_USB_STATUS_XACT_ERROR	The device returned a transaction error (defined for backward compatibility with the USB 1.0).
0xC0000012	IPMX_USB_STATUS_BABBLE_DETECTED	The device returned a babble detected error (defined for backward compatibility with the USB 1.0).
0xC0000013	IPMX_USB_STATUS_DATA_BUFFER_ERROR	Hardware status codes that range from 0x00000001 to 0x000000FF (defined for backward compatibility with the USB 1.0 stack).
0xC0000014	IPMX_USB_STATUS_NO_PING_RESPONSE	No response was received from the device for a ping packet sent by the Host.
0xC0000015	IPMX_USB_STATUS_INVALID_STREAM_TYPE	The stream type is invalid for the Endpoint.
0xC0000016	IPMX_USB_STATUS_INVALID_STREAM_ID	The stream identifier is invalid.
0xC0000030	IPMX_USB_STATUS_ENDPOINT_HALTED	A transfer was submitted to an Endpoint that is stalled.
0x80000200	IPMX_USB_STATUS_INVALID_URB_FUNCTION	Invalid URB function.
0x80000300	IPMX_USB_STATUS_INVALID_PARAMETER	Invalid parameter.
0x80000400	IPMX_USB_STATUS_ERROR_BUSY	The client driver caused an error by attempting to close an Endpoint, interface, or configuration handle with outstanding transfers.
0x80000500	IPMX_USB_STATUS_REQUEST_FAILED	The hub driver cannot complete a URB request.
0x80000600	IPMX_USB_STATUS_INVALID_PIPE_HANDLE	Invalid pipe handle.
0x80000700	IPMX_USB_STATUS_NO_BANDWIDTH	There was not enough bandwidth to open a requested Endpoint.
0x80000800	IPMX_USB_STATUS_INTERNAL_HC_ERROR	Unspecified Host controller error.
0x80000900	IPMX_USB_STATUS_ERROR_SHORT_TRANSFER	The transfer ended with a short packet, but the USBD_SHORT_TRANSFER_OK bit is not set for the pipe.
0xC0000A00	IPMX_USB_STATUS_BAD_START_FRAME	The requested start frame is not within the range of USBD_ISO_START_FRAME_RANGE frames of the current USB frame. Whenever this error occurs, the system sets the stall bit on the pipe.
0xC0000B00	IPMX_USB_STATUS_ISOCH_REQUEST_FAILED	The Host controller returns this error whenever all packets in an isochronous transfer complete with an error.
0xC0000C00	IPMX_USB_STATUS_FRAME_CONTROL_OWNED	The hub driver returns this error whenever the frame length control for the Host controller is being used by a driver other than the Host controller driver.
0xC0000D00	IPMX_USB_STATUS_FRAME_CONTROL_NOT_OWNED	The hub driver returns this error if the caller does not own frame length control and attempts to release or modify the Host controller frame length.
0xC0000E00	IPMX_USB_STATUS_NOT_SUPPORTED	The request was not supported.
0xC0000F00	IPMX_USB_STATUS_INVALID_CONFIGURATION_DESCRIPTOR	Invalid configuration descriptor.
0xC0001000	IPMX_USB_STATUS_INSUFFICIENT_RESOURCES	Insufficient resources.
0xC0002000	IPMX_USB_STATUS_SET_CONFIG_FAILED	An attempt to change the device configuration failed.
0xC0003000	IPMX_USB_STATUS_BUFFER_TOO_SMALL	The buffer is too small.
0xC0004000	IPMX_USB_STATUS_INTERFACE_NOT_FOUND	The interface was not found.

0xC0005000	IPMX_USB_STATUS_INAVLID_PIPE_FLAGS	Invalid pipe flags.
0xC0006000	IPMX_USB_STATUS_TIMEOUT	The request timed out.
0xC0007000	IPMX_USB_STATUS_DEVICE_GONE	The device is no longer present in the system.
0xC0008000	IPMX_USB_STATUS_STATUS_NOT_MAPPED	The device bus address is not mapped to system memory.
0xC0009000	IPMX_USB_STATUS_HUB_INTERNAL_ERROR	The device bus address is not mapped to system memory.
0xC0010000	IPMX_USB_STATUS_CANCELED	The USB stack reports this error whenever it completed a transfer because of an AbortPipe request from the client driver.
0xC0020000	IPMX_USB_STATUS_ISO_NOT_ACCESSED_BY_HW	The Host controller did not access the transfer descriptor (TD) that is associated with this packet. The USB stack reports this error in the packet status field of an isochronous transfer packet.
0xC0030000	IPMX_USB_STATUS_ISO_TD_ERROR	The Host controller reported an error in the transfer descriptor (TD). The USB stack reports this error in the packet status field of an isochronous transfer packet.
0xC0040000	IPMX_USB_STATUS_ISO_NA_LATE_USBPORT	The client driver submitted the packet on time, but the packet failed to reach the miniport driver on time. The USB stack reports this error in the packet status field of an isochronous transfer packet.
0xC0050000	IPMX_USB_STATUS_ISO_NOT_ACCESSED_LATE	The client driver did not submit the packet on time. The USB stack reports this error in the packet status field of an isochronous transfer packet.
0xC0100000	IPMX_USB_STATUS_BAD_DESCRIPTOR	Invalid descriptor.
0xC0100001	IPMX_USB_STATUS_BAD_DESCRIPTOR_BLEN	Invalid descriptor length.
0xC0100002	IPMX_USB_STATUS_BAD_DESCRIPTOR_TYPE	Invalid descriptor type.
0xC0100003	IPMX_USB_STATUS_BAD_INTERFACE_DESCRIPTOR	Invalid interface descriptor.
0xC0100004	IPMX_USB_STATUS_BAD_ENDPOINT_DESCRIPTOR	Invalid interface descriptor.
0xC0100005	IPMX_USB_STATUS_BAD_INTERFACE_ASSOC_DESCRIPTOR	Invalid interface association descriptor.
0xC0100006	IPMX_USB_STATUS_BAD_CONFIG_DESC_LENGTH	Invalid configuration descriptor length.
0xC0100007	IPMX_USB_STATUS_BAD_NUMBER_OF_INTERFACES	Invalid number of interfaces.
0xC0100008	IPMX_USB_STATUS_BAD_NUMBER_OF_ENDPOINTS	Invalid number of Endpoints.
0xC0100009	IPMX_USB_STATUS_BAD_ENDPOINT_ADDRESS	Invalid Endpoint address.
0xFFFFFFFF	IPMX_USB_STATUS_UNKNOWN_ERROR	Unknown error.

Table 19: Status code values