User Datagram Protocol for Universal MIDI Packets

Network MIDI 2.0 (UDP)
Transport Specification

MIDI Association Document: M2-124-UM

Document Version 1.0
Draft Date 2024-11-06

Published 2024-11-20

Developed and Published By
The MIDI Association
and
Association of Musical Electronics Industry (AMEI)

PREFACE

MIDI Association Document M2-124-UM
User Datagram Protocol for Universal MIDI Packets

The User Datagram Protocol for Universal MIDI Packets, described in this specification, provides a standardized way to transport MIDI Messages around IP based networks. This specification also defines methods for discovery and connecting to MIDI Devices located on the Network, as well as maintaining these connections.

http://www.amei.or.jp

https://www.midi.org

Version History

Table 1 Version History

Publication Date	Version	Changes
2024-11-20	1.0	Initial release

Contents

Figures

Tables

1 Introduction

1.1 Executive Summary

This specification defines the mechanisms and the message format for exchanging Universal MIDI Packet messages via a local area network. This enables the transport of MIDI 1.0 Protocol and MIDI 2.0 Protocol. This specification also defines discovery of available Network MIDI devices, and the management of Network MIDI Sessions.

Some examples of use-cases in this specification may include:

- Connectivity in home studios with multiple MIDI devices connected to a computer with Digital Audio Workstation software.
- Live performance with both networked multi-channel digital audio and a variety of MIDI instruments and controllers.
- Commercial facilities that include broadcast studios, recording studios, and universities with networked electronic music/piano labs.

This specification defines a way to connect those home studios or those performance venues via Ethernet and wireless LAN on a local area network.

Advantages of using a network for MIDI:

- Ethernet cables can transmit data up to 100 meters without any signal loss or degradation.
- Ethernet connections are electrically isolated, reducing the chances of electrical grounding noise issues.
- The connection topology can be changed in software without having to move physical cables.
- Peer to peer connection is possible without the need to route data through a computer.

Many transports for audio use Ethernet. Some of those are open standards (i.e. AES67) and some are proprietary. Being able to run MIDI 2.0 as a control protocol over the same cables that run audio can greatly expand the use of MIDI 2.0. MIDI 2.0 expands outside the limitations of MIDI 1.0, now using packets comprised of 32-bit words, no bandwidth limitations, and room for thousands of new messages.

A MIDI 2.0 UDP transport has many benefits for both end users and companies that make MIDI and audio products.

1.2 Background

This specification defines the methods where two (or more) MIDI Devices can discover and connect across an IP based Network and exchange UMP messages. This network protocol is targeted to run on Local Area Networks with a focus on Ethernet and wireless LAN (IEEE802.11). Other IP transports like UDP over Bluetooth or the Internet are not supported by this specification.

Discover MIDI Devices using DNS-SD

This specification defines the use of the DNS-SD ***[RFC6763]*** standard for discovery. DNS-SD builds on mDNS ***[RFC6762]*** and provides a common way for MIDI Devices to publish their network related information. This information includes the UMP Endpoint Name, the Product Instance Id, as well as the IP and UDP port number to connect and establish a Session.

UDP Message Structure and Commands

This specification defines the use of UDP for establishing a Session and sending UMP data between two MIDI Devices. Each UDP packet contains one or more Commands to either send UMP data or manage the Session.

1.3 References

1.3.1 Normative References

[EIA01] ***Control Network Protocol Specification***, October 1999, EIA-709.1-A

[RFC768] ***User Datagram Protocol (UDP),*** RFC 768, https://datatracker.ietf.org/doc/html/rfc768

[RFC791] ***Internet Protocol (IP),*** RFC 791, https://datatracker.ietf.org/doc/html/rfc791

[RFC3927] ***Dynamic Configuration of IPv4 Link-Local Addresses***, RFC 3927, https://tools.ietf.org/html/rfc3927

[RFC6762] ***Multicast DNS (mDNS),*** RFC 6762, https://tools.ietf.org/html/rfc6762

[RFC6763] ***DNS-Based Service Discovery,*** RFC 6763, https://tools.ietf.org/html/rfc6763

[MA01] ***Complete MIDI 1.0 Detailed Specification***, Document Version 96.1, Third Edition, Association of Musical Electronics Industry, http://www.amei.or.jp/, and The MIDI Association, https://www.midi.org/

[MA02] ***M2-100-U MIDI 2.0 Specification Overview***, Version 1.1, Association of Musical Electronics Industry, http://www.amei.or.jp/, and The MIDI Association, https://www.midi.org/

[MA03] ***M2-101-UM MIDI Capability Inquiry (MIDI-CI)***, Version 1.2, Association of Musical Electronics Industry, http://www.amei.or.jp/, and The MIDI Association, https://www.midi.org/

[MA04] ***M2-102-U Common Rules for MIDI-CI Profiles***, Version 1.1, Association of Musical Electronics Industry, http://www.amei.or.jp/, and The MIDI Association, https://www.midi.org/

[MA05] ***M2-103-UM Common Rules for Property Exchange***, Version 1.1, Association of Musical Electronics Industry, http://www.amei.or.jp/, and The MIDI Association, https://www.midi.org/

[MA06] ***M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol***, Version 1.1.2, Association of Musical Electronics Industry, http://www.amei.or.jp/, and The MIDI Association, https://www.midi.org/

[USBIF01] ***USB Class Specification for MIDI Devices***, Version 1.0, USB Implementers Forum, https://www.usb.org/

[USBIF02] ***USB Class Specification for MIDI Devices***, Version 2.0, USB Implementers Forum, https://www.usb.org/

1.3.2 Informative References

No Informative References

1.4 Terminology

1.4.1 Definitions

AMEI: Association of Musical Electronics Industry. Authority for MIDI Specifications in Japan.

Chunk: A single System Exclusive message that is one segment of a complete Property Exchange message which spans multiple System Exclusive messages.
Client: A MIDI 2.0 Device that connects to a Host on the Network

Command: A code that informs the receiver of how it should process the Command Packet

Command Packet: A set of data inside a UDP Packet that has some instruction for the receiver. A UDP packet may contain multiple Command Packets.

CryptoNonce: A cryptographic nonce is an arbitrary string used only once during authenticated Session initiation. It is used for preventing replay attacks of an Invitation with Authentication.

Destination: A Receiver to which the Sender intends to send MIDI messages.

Device: An entity, whether hardware or software, which can send and/or receive MIDI messages.

DNS: Domain Name System. A protocol on application layer used for resolving IP addresses and other information from a given host name.

DNS-SD: DNS Service Discovery, defines the records for publishing device information. Usually used with mDNS. DNS-SD is specified in RFC 6763 ***[RFC6763]***.

Host: An entity of a MIDI 2.0 Device that exposes a UMP Endpoint to the Network

Inquiry: A message sent by an Initiator to begin a Transaction.

Invitation: The action taken to request a Session via the network.

IP: Internet Protocol, often used as a short form for IP version 4 address [***RFC791***].

MA: MIDI Association. Authority for MIDI specifications worldwide except Japan. See also MMA.

mDNS: multicast DNS, used for resolving IP addresses to host names and providing additional information in small local networks. mDNS is often used in conjunction with DNS-SD. mDNS is specified in RFC 6762 ***[RFC6762]***.

MIDI 1.0 Protocol: Version 1.0 of the MIDI Protocol as originally specified in ***[MA01]*** and extended by MA and AMEI with numerous additional MIDI message definitions and Recommended Practices. The native format for the MIDI 1.0 Protocol is a byte stream, but it has been adapted for many different transports. MIDI 1.0 messages can be carried in UMP packets. The UMP format for the MIDI 1.0 Protocol is defined in the M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification ***[MA06]***.

MIDI 1.0 Specification: Complete MIDI 1.0 Detailed Specification, Document Version 96.1, Third Edition ***[MA01]***.

MIDI 2.0: The MIDI environment that encompasses all of MIDI 1.0, MIDI-CI, Universal MIDI Packet (UMP), MIDI 2.0 Protocol, MIDI 2.0 messages, and other extensions to MIDI as described in AMEI and MA specifications.

MIDI 2.0 Protocol: Version 2.0 of the MIDI Protocol. The native format for MIDI 2.0 Protocol messages is UMP as defined in M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification ***[MA06]***.

MIDI-CI: MIDI Capability Inquiry ***[MA03]***, a specification published by The MIDI Association and AMEI.

MIDI Endpoint: A Device which is an original source of MIDI messages or final consumer of MIDI messages. Supports only MIDI 1.0 or is switchable between MIDI 1.0 and MIDI 2.0 Protocol messages.

MIDI Manufacturers Association: A California nonprofit 501(c)6 trade organization, and the legal entity name of the MIDI Association.

MIDI Transport: A hardware or software MIDI connection used by a Device to transmit and/or receive MIDI messages to and/or from another Device.

MMA: See MIDI Manufacturers Association.

MUID (MIDI Unique Identifier): A 28-bit random number generated by a Device used to uniquely identify the Device in MIDI-CI messages sent to or from that Device.

Product Instance Id: a statistically unique ASCII identifier for a unit of a Device . See ***[MA06].***

Protocol: There are two defined MIDI Protocols: the MIDI 1.0 Protocol and the MIDI 2.0 Protocol, each with a data structure that defines the semantics for MIDI messages. See ***[MA01]*** and ***[MA06]***.

Receiver: A MIDI Device which has a MIDI Transport connected to its MIDI In.

Renderer: A Receiver which recognizes and acts upon any of the MIDI messages that it receives.

Sender: A MIDI Device which transmits MIDI messages to a MIDI Transport which is connected to its MIDI Out or to its MIDI Thru port.

Session: A Client connecting to a Host establishes a Session. The Session exists until such time as either end sends a Command to terminate the Session, or the connection is lost forcing termination.

Session State: One of six possible states that any Client or Host can be in.

Sequence Number: An incrementing number field used for declaring and maintaining the sequential order of UMP packets and for data integrity mechanisms.

Source: A Source is a Sender which originates or generates MIDI messages. A Source does not include a Sender which is retransmitting messages which originated in another MIDI Device.

Transaction: An exchange of MIDI messages between two MIDI Devices with a bidirectional connection. All the MIDI messages in a single Transaction are associated and work together to accomplish one function. The simplest Transaction generally consists of an inquiry sent by one MIDI Device and an associated reply returned by a second MIDI Device. A Transaction may also consist of an inquiry from one MIDI Device and several associated replies from a second MIDI Device. A Transaction may be a more complex set of message exchanges, started by an initial inquiry from one MIDI Device and multiple, associated replies exchanged between the first MIDI Device and a second MIDI Device.

UDP: User Data Protocol ***[RFC768]***

UMP: Universal MIDI Packet, see ***[MA06]***.

UMP Endpoint: A MIDI Endpoint which uses the UMP Format.

UMP Endpoint Name: The name to be presented to the user for a UMP Endpoint.

UMP Format: Data format for fields and messages in the Universal MIDI Packet, see ***[MA06]***.

UMP MIDI 1.0 Device: any Device that sends or receives MIDI 1.0 Protocol messages using the UMP ***[MA06]***. Such Devices may use UMP Message Types that extend the functionality beyond Non-UMP MIDI 1.0 Systems.

Universal MIDI Packet (UMP): The Universal MIDI Packet is a data container which defines the data format for all MIDI 1.0 Protocol messages and all MIDI 2.0 Protocol messages. UMP is intended to be universally applicable, i.e., technically suitable for use in any transport where MA/AMEI elects to officially support UMP. For detailed definition see M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification ***[MA06]***.

USB Endpoint: The source or sink of data sent over USB, as defined in the USB core specifications.

USB-MIDI Endpoint: A USB Endpoint used to transfer MIDI Data as defined in the USB Class Specification for MIDI Devices v1.0 ***[USBIF01]***.

1.4.2 Reserved Words and Specification Conformance

In this document, the following words are used solely to distinguish what is required to conform to this specification, what is recommended but not required for conformance, and what is permitted but not required for conformance:

Table 2 Words Relating to Specification Conformance

Word	Reserved For	Relation to Specification Conformance
shall	Statements of requirement	**Mandatory** A conformant implementation conforms to all 'shall' statements.
should	Statements of recommendation	**Recommended but not mandatory** An implementation that does not conform to some or all 'should' statements is still conformant, providing all 'shall' statements are conformed to.
may	Statements of permission	**Optional** An implementation that does not conform to some or all 'may' statements is still conformant, providing that all 'shall' statements are conformed to.

By contrast, in this document, the following words are never used for specification conformance statements; they are used solely for descriptive and explanatory purposes:

Table 3 Words Not Relating to Specification Conformance

Word	Reserved For	Relation to Specification Conformance
must	Statements of unavoidability	Describes an action to be taken that, while not required (or at least not directly required) by this specification, is unavoidable. Not used for statements of conformance requirement (see 'shall' above).
will	Statements of fact	Describes a condition that as a question of fact is necessarily going to be true, or an action that as a question of fact is necessarily going to occur, but not as a requirement (or at least not as a direct requirement) of this specification. Not used for statements of conformance requirements (see 'shall' above).
can	Statements of capability	Describes a condition or action that a system element is capable of possessing or taking. Not used for statements of conformance permission (see 'may' above).
might	Statements of possibility	Describes a condition or action that a system element is capable of electing to possess or take. Not used for statements of conformance permission (see 'may' above).

2 UDP Transport Overview

This network protocol is targeted to run on Local Area Networks with a focus on Ethernet and wireless LAN (IEEE802.11). Other IP transports like UDP over Bluetooth or the Internet are not supported by this specification.

For discovery, mDNS is used, following DNS-SD conventions. UDP packets are used to establish, maintain, and terminate Sessions and are also used to transfer UMP messages. The use of mDNS is optional but recommended and it is the only mechanism defined in this specification for automated discovery.

2.1 UDP Host

A MIDI 2.0 UDP Host (Host) is any MIDI 2.0 Device that exposes a UMP Endpoint to the Network. A Device may expose any number of Host UMP Endpoints by reserving a different UDP port for each one. Each Host UMP Endpoint handles its own authentication and may handle any number of MIDI 2.0 UDP Clients as it has resources for.

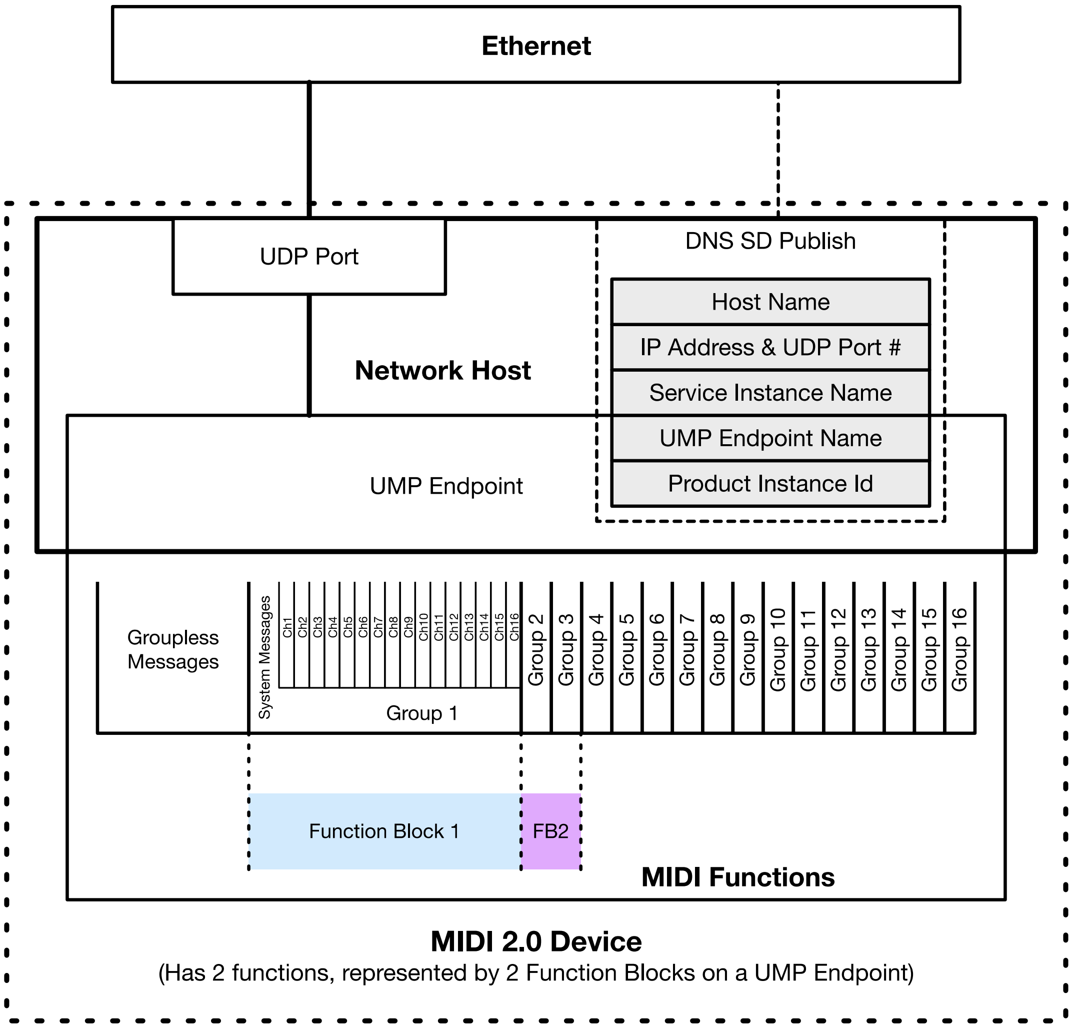

Figure 1 UMP Endpoint Network Host

For Devices with multiple UMP Endpoints, see Section 9 for more information.

2.2 UDP Client

A Client is a UMP Endpoint that initiates a session with a Host. A Client shall use a unique UDP port to communicate with the Host. This Client port will transmit and receive UDP packets to and from Host for bidirectional communication. A Client shall not run multiple sessions to the same Host from the same UDP port. In order to start a second session to a Host, a Client shall use a different UDP port.

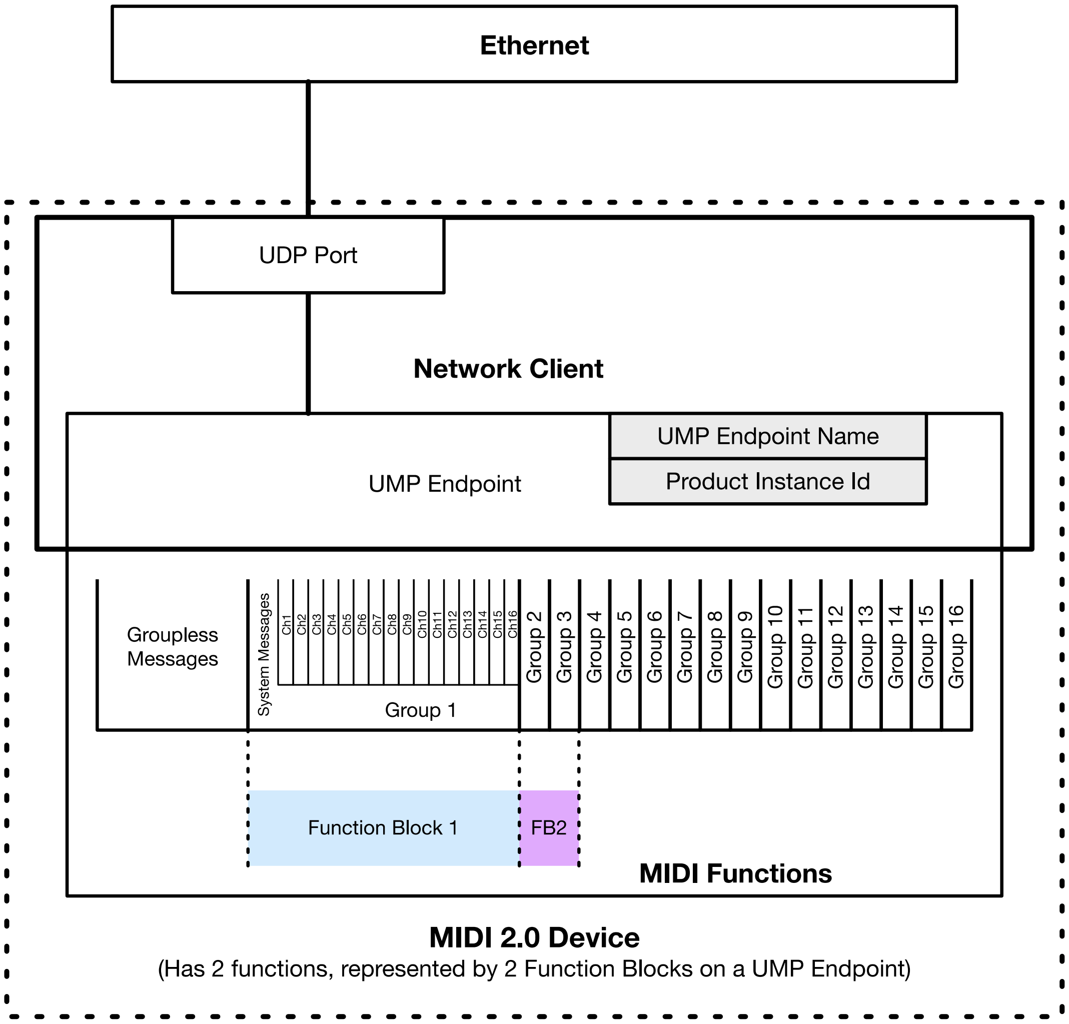

Figure 2 UMP Endpoint Network Client

Devices may implement both Host role and Client role at the same time. This allows the Device to be discoverable and reachable by other Clients while also allowing the Device to initiate a session with other Hosts.

2.3 Data Integrity

The specification defines two optional mechanisms to help data integrity.

1. A Forward Error
2. Correction mechanism is defined to provide a simple implementation which resends recently sent data with any new message.
3. A Retransmit mechanism is defined for use when the Sender has the ability to respond to missing data requests from the Receiver.

These two mechanisms rely on an incrementing Sequence Number which is included in packets which transfer Universal MIDI Packet data.

2.4 Wireless LAN Performance

Using wireless LAN for real-time messages can lead to compromised quality of service compared to Ethernet. Wireless LAN shares frequencies with other wireless technologies and other wireless LAN traffic. Walls, people, and other obstacles can reduce wireless LAN reception quality, and various other radio interferences can lead to interruptions. Interruptions that may affect the MIDI transmission can vary from a few milliseconds to significantly longer. The wireless LAN implementation may provide limited error recovery. Even with error recovery wireless LAN remains prone to missing, duplicated, and out of order UDP packets. A Network UDP implementation should expect these error cases and leverage the data integrity facilities in this specification. See Section *7.2*.

2.5 Network Security

Even in the scope of a Local Area Network, it is important to understand that the defined protocol is not encrypted or secure, so an attacker can easily eavesdrop on UMP streams. Although the Session initiation allows for user confirmation or password authentication, techniques like man-in-the-middle attacks and replay attacks allow an intruder to fake a session initiation and get Session access.

3 IP Addresses and Port Numbers

3.1 IP Address Assignment

For UMP over IP networks, each Device shall have at least one IPv4 address. Each Device should also support IPv6. For UMP over UDP, the device shall support at least one of the following:

1. A way for the user to set a static IP address + netmask
2. A DHCP client and/or a DHCP server + Link Local address auto-assignment as fallback, as defined in RFC 3927 ***[RFC3927]***.

Devices should offer both options.

A Host with active Discovery (see Section ***4***) shall expose its IP address(es) via the A (and AAAA) records.

3.2 Host UDP Port Assignment

Every Host shall allocate one dedicated UDP port during the lifetime of the Host. The Host listens on this port for incoming UDP packets from remote Client instances. The Host shall use this UDP port as the source port when sending UDP packets to Clients. A Host shares its UDP port with all Clients.

A Host with active Discovery via DNS-SD shall expose the Host UDP port number via the SRV record (See chapter 4 Discovery).

When serving multiple Clients, the Host shall uniquely identify the connection for each Client via the Client's source IP address and UDP port number of the incoming UDP packets.

If a Device implements multiple Hosts, then it shall allocate a UDP port for each Host instance.

3.3 Client UDP Port Assignment

Every Client shall allocate one dedicated UDP port. The Client uses this UDP Port as a source port when sending UDP packets to a Host, and it also receives UDP packets from the Host on this port.

Clients do not expose the UDP Ports using DNS-SD Discovery. Clients should allocate a free UDP port number just before connecting to a Host. After the Session with the Host has been terminated, the Client may deallocate the UDP port. Clients may use a new UDP port number for every Session with a Host.

Resource-constrained Devices may use a single UDP port for multiple Clients. A Client shall not connect to the same Host multiple times simultaneously using the same source UDP port.

Devices which act both as a Host and as a Client, should not share a single UDP port for Host and Client. If a Device uses the same UDP port for both Host and Client, then the Device's Client is unable to connect to a Host with which a Session is already established.

4 Discovery

4.1 Overview

If Discovery is enabled, then mDNS with DNS-SD shall be used to discover Hosts. UMP Endpoints may also establish sessions to other UMP Endpoints directly without automatic discovery (e.g., by entering the IP address and port number directly).

Following the DNS-SD standard ***[RFC6763]***, the following information is provided by each available Host via mDNS:

1. PTR: the mDNS Service Instance Name
2. SRV: the UDP port (for session and data protocol) and the mDNS hostname
3. TXT:
   A. UMP Endpoint Name
   B. Product Instance Id
4. A: one or more IPv4 address(es)
5. AAAA: one or more IPv6 address(es)

All other information is handled via the Session Protocol and via UMP data.

User applications may connect to OS-provided mDNS services to advertise a Host service, or to implement their own mDNS service. Developers should follow any guidance from the individual operating system companies, especially if the operating system company recommends using a MIDI-specific API for this transport.

4.2 DNS Pointer (PTR) Record

Each PTR record lists one service instance for the given service type. A Device may advertise multiple service instances using multiple PTR records.

Table 4 PTR Fields

Field	Value	Description
Service Type	_midi2._udp.local	The registered service type in the local domain.
Service Instance Name	<string>	An arbitrary name used for discovery only.

Example of a complete PTR record:

```
_midi2._udp.local.  PTR   173ACE-MIDIWorkbench._midi2._udp.local.
```

Service Instance Name

The Service Instance Name is an internal identifier for one Host instance. See DNS-SD ***[RFC6763]*** for more information. If a Device exposes multiple MIDI 2.0 Host instances, the Device shall ensure that the Service Instance Name for each Host is unique for the given hostname. Service Instance Names should be persistent across power cycles.

One way to construct the Service Instance Name is to use the Product Instance Id (see Section ***4.4***), or parts of it, followed by a dash and the model name. See the DNS-SD specification ***[RFC6763]*** for information on the character and length limitations of the Service Instance Name.

The Service Instance Name should not be displayed to the user. The UMP Endpoint Name provided in the TXT record should be used for displaying this Host to the user.

4.3 DNS Service (SRV) Record

The SRV record provides the UDP port number and the mDNS hostname for a given service instance name. The maximum length of the SRV record is 255 bytes.

For MIDI 2.0 Host discovery, the following fields are used:

Table 5 SRV Fields

Field	Value	Comments
Service Instance Name	<string>	Service Instance name of the Host, as referenced by the PTR mDNS record.
Service Type	_midi2._udp.local.	the registered service type for MIDI 2.0 services
Port	16-bit number	The UDP port number of this service on the Host.
Hostname	<string>	The mDNS hostname (ASCII), to be resolved using A and AAAA records. Must include the .local. domain.

Scheme:

```
<Service Instance>._midi2._udp.local. <TTL> IN SRV 0 0 <Port> <Host>
```

Example for a complete SRV record:

```
173ACE-MIDIWorkbench._midi2._udp.local. 60 IN SRV 0 0 5673 desktop.local.
```

Hostname

The hostname is an identifier of the device, which can be resolved to one or more IP addresses using the A and AAAA records. The hostname consists of ASCII characters only. The hostname shall include the `.local` domain. Hostnames shall be unique in a network. Devices should provide a means to let the user configure the hostname. Devices without a GUI should include a unique Id in the Hostname, for example the Product Instance Id (see Section *4.4*), or a part of the Product Instance Id, so that multiple units of the same model will use different hostnames. It is highly recommended that the hostname is persistent across power cycles.

4.4 DNS Text (TXT) Record

Every Device shall include a TXT record which provides additional information about a given Service Instance. Each label in the TXT record defines a key/value pair, using the equals sign = as delimiter.

The (theoretical) maximum length of each key/value pair is 255 bytes, consisting of <key>, "=", and <value>. However, the DNS-SD specification specifies that the TXT record is intended to be small, ideally under 200 bytes. Furthermore, to be consistent with the UMP specification, the values have additional restrictions as specified below.

Two keys are defined for the TXT record:

Table 6 TXT Keys

TXT Key	Description
UMPEndpointName	The UMP Endpoint Name of the Host in (UTF-8, up to 98 bytes).
ProductInstanceId	A unique Id for the device of this Host as also reported by the UMP Product Instance Id Notification message (ASCII ordinal 32-126, up to 42 bytes).

Example for a TXT record:

```
173ACE-MIDIWorkbench._midi2._udp.local. TXT
    UMPEndpointName=MIDI Workbench
    ProductInstanceId=173ACEHDI3nVAUiFH
```

Operating systems and devices may use the UMPEndpointName and ProductInstanceId to recall Device properties when reconnecting to devices.

UMPEndpointName

The UMP Endpoint Name TXT field represents the name of the Host which should be displayed to the user. The UMP Endpoint Name TXT field shall be the same name as the UMP Endpoint Name used in the Invitation Reply Commands, and it shall also be the same as reported by the UMP Endpoint Name Notification message. To be consistent with the UMP specification, the UMP Endpoint Name shall be encoded in UTF-8, and the length shall not exceed 98 bytes. See the UMP and MIDI 2.0 Protocol specification ***[MA06]***.

A Device with multiple Host instances shall use a different UMP Endpoint Name for each Host instance.

ProductInstanceId

The Product Instance Id shall be a set of ASCII characters in the ordinal range 32-126 only.

This Product Instance Id reported here shall be the same as reported by a Product Instance Id Notification message. To be consistent with the UMP specification, the Product Instance Id length should not exceed 42 bytes. See the UMP and MIDI 2.0 Protocol specification ***[MA06]***.

A Device with multiple Host instances should use the same Product Instance Id for all Hosts.

4.5 A/AAAA Records

The A and AAAA records resolve the host name, as given in the SRV record, to an IP address. Multiple A and/or AAAA records can be used to specify multiple IP addresses for the same host.

Example for A and AAAA records:

```
desktop.local.  A  192.168.1.6
desktop.local.  A  192.168.76.1
desktop.local.  AAAA  fe80::ca8:bfe7:dbaf:7820
```

4.6 Time-To-Live (TTL)

Specifications for mDNS/DNS-SD (***[RFC6762]*** and ***[RFC6763]***) define how a Host un-publishes its mDNS service when it is being terminated. However, there are circumstances, such as a cable disconnect or a system crash, where this does not occur. Therefore, a Host should not set its service TTL longer than one minute. This allows devices on the network to remove stale entries for mDNS service for Hosts that have been disconnected without unpublishing the mDNS service.

5 UDP Payload Format

5.1 Overview

All UDP packets are sent peer to peer from Host to Client or from Client to Host.

The UDP Packet always starts with a 4-byte Signature, followed by Command Packets of variable size.

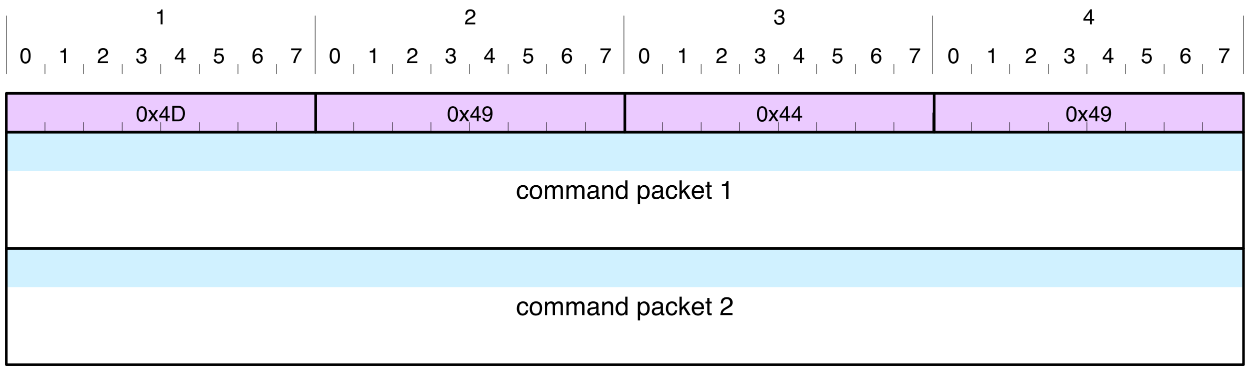

Figure 3 Example for UDP Packet with two Command Packets

5.1.1 Maximum UDP Packet Size

In order to avoid packet fragmentation on the underlying transports, UDP packets should not exceed 1400 bytes. Implementations should take great care to avoid fragmented UDP packets. Fragmentation can cause higher latency and missing fragments. Furthermore, certain embedded UDP stacks do not support fragmentation and will simply not process fragmented UDP packets.

5.2 Signature

All UDP packets shall start with the value 0x4D494449 ("MIDI" in ASCII). This 32-bit word is only used once per UDP packet. A receiver shall verify this first 32-bit word. If this verification fails, then the rest of the packet shall be ignored.

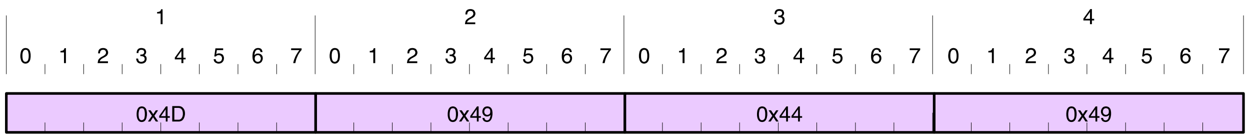

Figure 4 UDP Payload Header - Signature

5.3 Command Packet Data Format

Command Packets manage Sessions and send the UMP Data packets. Common properties of all Command Packets are outlined below.

Number Encoding

All numbers and UMP data in Command Packets are encoded using network byte order (“big endian”). If not otherwise mentioned, they are unsigned.

Flags

Fields of type Flags are encoded bit wise, where each bit corresponds to a specific function or meaning.

Enum

A field that encodes information for specific values.

String Encoding

All strings in Command Packets shall be UTF-8 encoded, without a Byte Order Mark, except where a different encoding is specified. If the text ends in the middle of a Command word, then the remaining data bytes shall be set to 0x00 to indicate the end of the text.

Receivers shall accept a string which is bounded by the Command Packet length. An empty string at the end of a Command may be omitted.

Reserved Bytes

All fields, bytes, and bits in a Command Packet that are specified as “Reserved” shall be set to zero. Receivers shall ignore all Reserved fields, bytes, and bits even if they are not zero.

Extra Data

Upon reception of a Command, any payload words following the defined fields shall be ignored, as long as the additional payload words are accounted for in the Payload Length field. Future versions of this specification may define optional extensions.

5.4 Command Packet Header and Payload

Every Command Packet starts with a 32-bit header. Immediately following the header is the Command Payload.

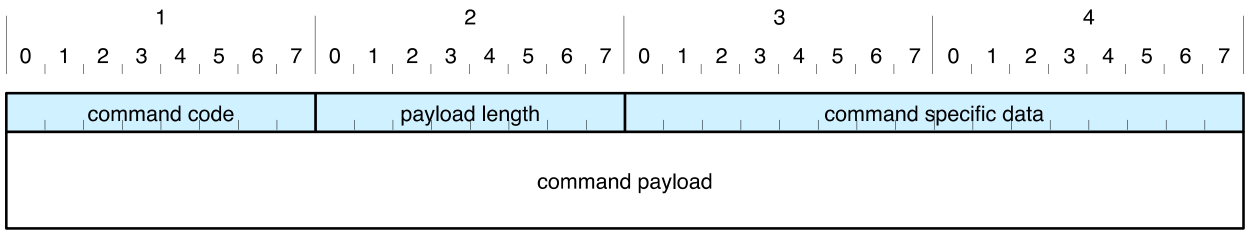

Figure 5 Command Packet Structure

Table 7 Command Packet Header Fields

Field	Size (bytes)	Description
Command Code	1	
Command Payload Length	1	Length in number of 32-bit words of the Command Payload which follows the Command Packet Header
Command Specific Data	2	

Command Code

The Command Code is used to determine the action or declare the Payload data type.

Command Payload Length

The length of the Command Payload is the number of 32-bit words. The Command Payload starts directly after the 32-bit header. If the Command Payload Length is zero, then this Command Packet does not use any Payload and the total length of the Command Packet is 32-bits, i.e. just the header.

Command Specific Data

16-bit field to represent extended data about the Command Packet. The Command Specific Data field is present in every Command Packet. If not used by a Command, the Command Specific Data shall be set to 0.

Some Commands split the Command Specific Data into Data 1 and Data 2, each 8 bit. Command Specific Data 1 corresponds to the high byte of the 16-bit Command Specific Data.

Command Payload

Immediately following the Command Packet Header is the actual payload for the given command. The payload is of variable length given by the Payload Length field in the header. The data format of the payload depends on the respective command code. The payload size is always a multiple of 4 bytes, because the length is specified as the number of 32-bit words.

To see an example of a UMP Data Command, see Section ***7.1 UMP Data***.

5.5 Command Codes and Packet Types

The following table lists all available Command Codes with links to the respective definition.

Table 8 List of all Commands

Command Code	Name	Command Specific Data	Command Payload	Direction between Host (H) and Client (C)	Section
0xFF	UMP Data	Sequence Number	UMP Data	both	***7.1***
0x01	Invitation	Capabilities	UMP Endpoint Name + Product Instance Id	C → H	***6.4***
0x02	Invitation with Authentication		sha256 digest	C → H	***6.9***
0x03	Invitation with User Authentication		sha256 digest +username,	C → H	***6.10***
0x10	Invitation Reply: Accepted		UMP Endpoint Name + Product Instance Id	H → C	***6.5***
0x11	Invitation Reply: Pending		UMP Endpoint Name + Product Instance Id	H → C	***6.6***
0x12	Invitation Reply: Authentication Required		Crypto-Nonce	H → C	***6.7***

0x13	Invitation Reply: User Authentication required		Crypto-Nonce	H → C	***6.8***
0x20	Ping		Ping Id	both	***6.13***
0x21	Ping Reply		Ping Id	both	***6.14***
0x80	Retransmit Request	requested Seq. Number	none	both	***7.2.3***
0x81	Retransmit Error	requested Seq. Number	none	both	***7.2.4***
0x82	Session Reset		none	both	***6.11***
0x83	Session Reset Reply		none	both	***6.12***
0x8F	NAK	NAK Code Command Code	Cmd Hdr, Opt. Command	both	***6.15***
0xF0	Bye	Bye Code	Opt. Command	both	***6.16***
0xF1	Bye Reply		none	both	***6.17***

Unknown Command Codes

Upon reception of a Command with an unknown Command Code, the receiver shall reply with a NAK Command with code "Command not supported".

When receiving a NAK Command with code "Command not supported", the endpoint should not send the respective Command again.

Examples for UDP Packets containing Commands

For examples of completed UDP packets containing Network MDI 2.0 (UDP) Commands, see Appendix ***A.1***.

5.6 Sequence Numbers

The Sequence Number, in the UMP Data Command and both Retransmit Commands, is used for ordering UMP packets transferred using the UMP Data Command. The Sequence Number is a 16-bit number in the range 0x0000 to 0xFFFF. The Sequence Number is increased for every UMP Data Command. The value wraps around to 0x0000 after 0xFFFF. See Section *7.1*.

On reception, the Sequence Number can be used to detect duplicate, unordered, and missing UMP Data Commands. The mechanisms for data stream integrity use the Sequence Number to recover from transmission problems. For more information, see Section *7.2*, Section *7.2.2,* and Section *7.2.3*.

6 Session Commands

6.1 Session States

There are 6 states for any Client / Host relationship:

- **Idle** - Device may be aware of each other (e.g. through mDNS), however neither device has sent an Invitation. The two Devices are not in a Session.
- **Pending Invitation** - Client has sent Invitation, however the Host has not accepted the Invitation. A Bye Command has not been sent or received.
- **Authentication Required** - Host has replied with Invitation Reply: Authentication Required or Invitation Reply: User Authentication Required. No timeout has yet occurred. This state is different from Idle, because the Client and Host need to store the CryptoNonce.
- **Established Session** - Invitation accepted, UMP Commands can be exchanged.
- **Pending Session Reset** - One Device has sent the Session Reset Command and is waiting for Session Reset Reply, and a timeout has not yet occurred.
- **Pending Bye** - one Endpoint has sent Bye and is waiting for Bye Reply, and a timeout has not yet occurred.

The following table lists the Commands which are allowed in each of the Session States.

Table 9 List of valid Commands per Session State

Session State	Valid Commands
Every State	NAK Ping Ping Reply Bye
Idle	Invitation
Pending Invitation	Invitation Reply: Accepted Invitation Reply: Pending Invitation Reply: Authentication Required Invitation Reply: User Authentication required
Authentication Required	Invitation with Authentication Invitation with User Authentication
Established Session	UMP Data Retransmit Request Retransmit Error
Pending Session Reset	Session Reset Reply
Pending Bye	Bye Reply

Note: If a Receiver receives a Command which is not appropriate to the current Session State, the expected response, such as a NAK or a Bye, depends on the Command. See the section for each Command for the defined response.

6.2 Repeated Commands

Some Commands should be repeated until the respective reply is received or until a timeout occurs. A reasonable delay should be applied before sending the repeated Command. A delay between 300ms and 2 seconds is recommended.

If a Device reaches its preferred timeout without receiving a suitable reply, then the Device shall cease repeating the Command and send a Bye Command (except for a repeated Bye Command). See Section ***6.16***.

Commands which should be repeated:

- Invitation
- Invitation with Authentication
- Invitation with User Authentication
- Session Reset
- Bye

6.3 Session Life Cycle

6.3.1 Standard Session

The following diagram shows the Commands involved in establishing a Session.

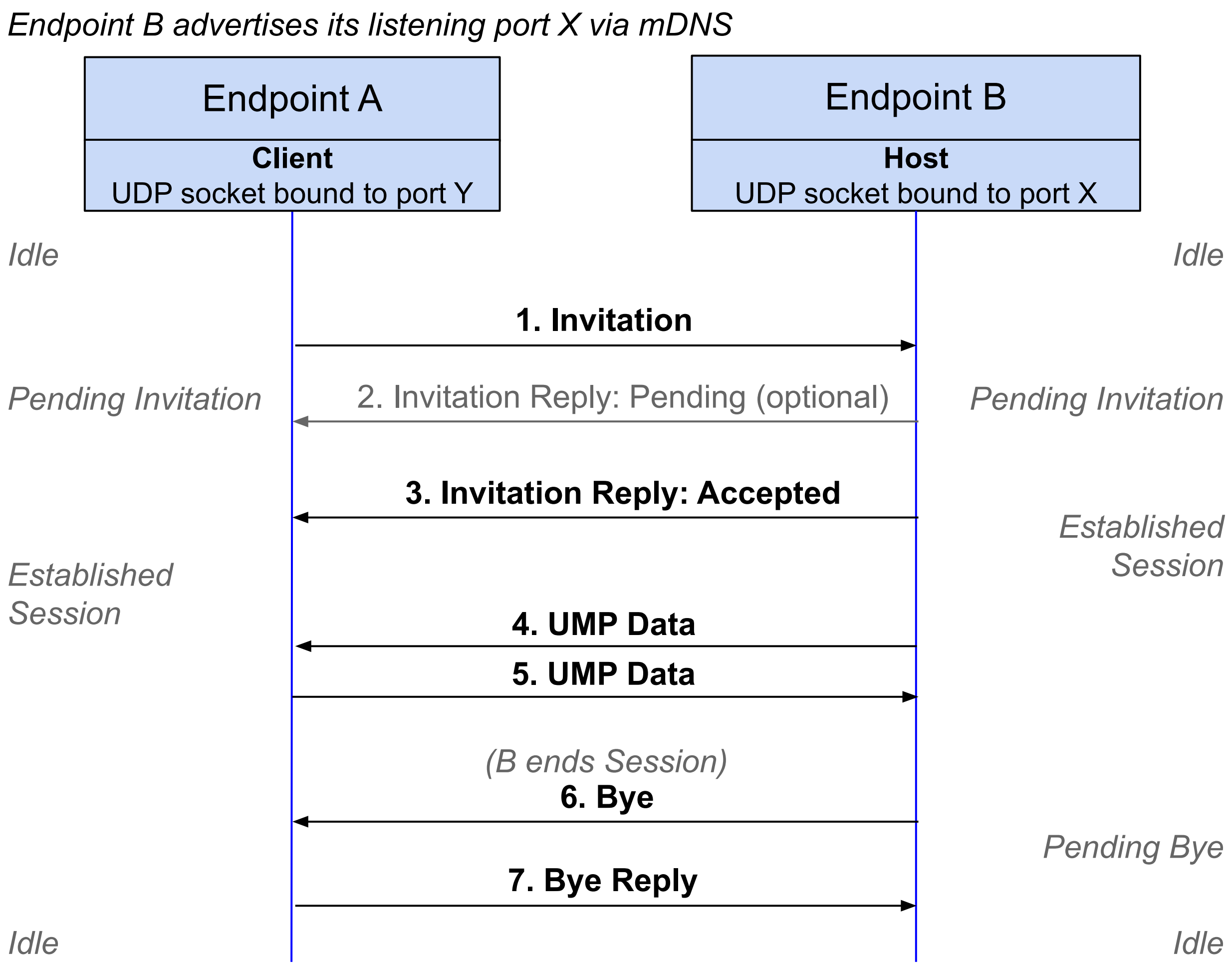

Figure 6 Establishing a Session

6.3.2 Session With Authentication

The following diagram shows the Commands involved in establishing a Session with Authentication.

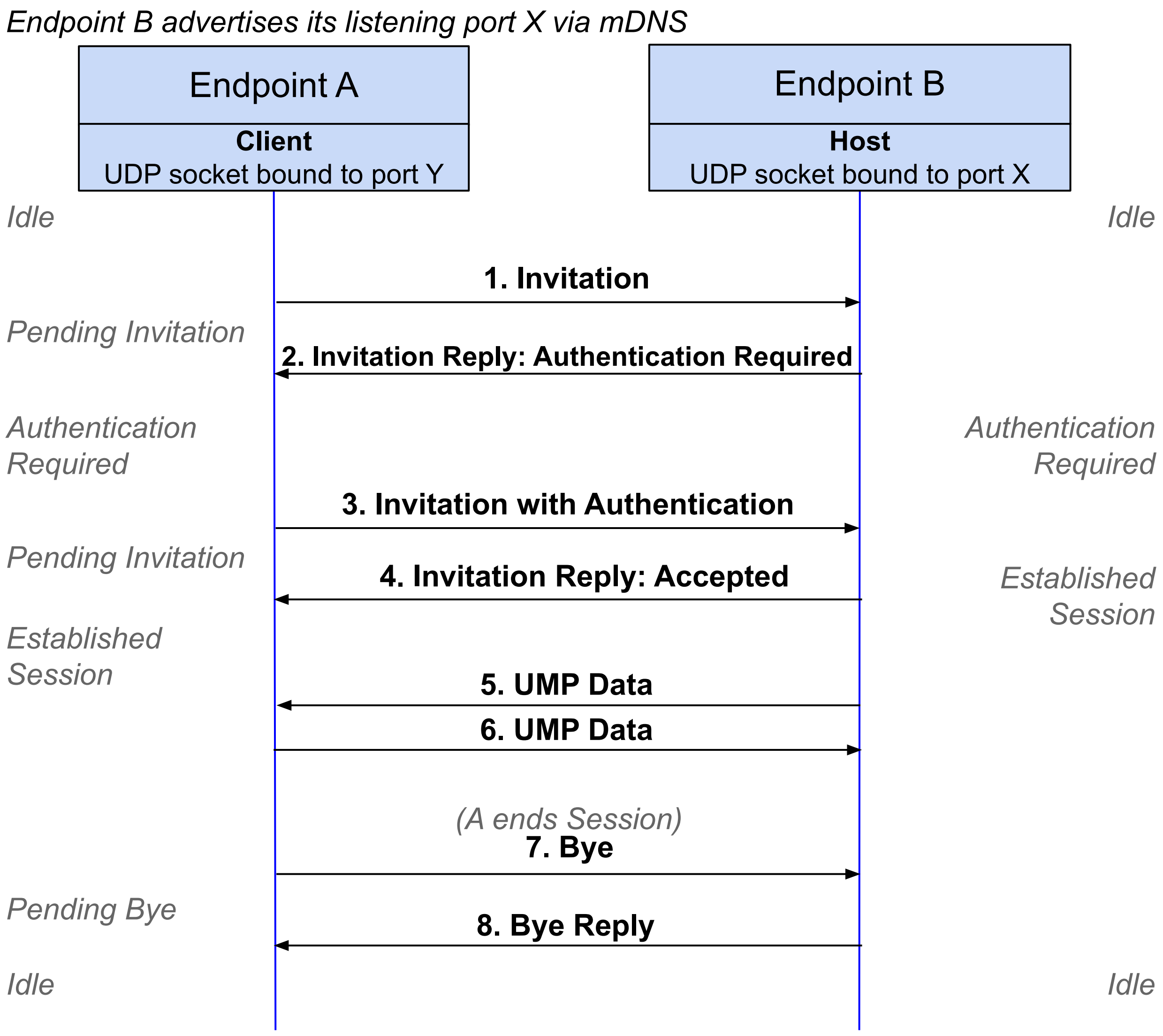

Figure 7 Establishing Session with Authentication

6.4 Invitation

The Client shall send an Invitation Command to a Host to initiate a Session.

After sending this Command, the Invitation is pending until a Bye Command or Invitation Reply is received. Either side may terminate a pending Invitation with a Bye Command.

The Invitation Command should be sent repeatedly, with a reasonable delay between Invitation Commands, until a reply is received. If the Client does not receive a reply for a reasonable time, it may apply a time-out. If the Client considers the Invitation failed, the Client shall terminate the Invitation with a Bye Command with reason 0x80 (Invitation Canceled).

If a Host receives an Invitation Command, the Host shall respond with one of the following:

- accept the invitation by replying with Invitation Reply: Accepted. See Section ***6.5***.
- reject the invitation and end the session with Bye Command with a suitable reason, for example 0x40 (Invitation failed: too many opened sessions). See Section ***6.16***.
- reject the invitation with Invitation Reply: Authentication Required (See Section ***6.7***) or with Invitation Reply: User Authentication Required (See Section ***6.8***) because an authentication scheme is required
- Optionally: notify the Client that permission is being requested with an Invitation Reply Pending Command (see Section ***6.6***). An Invitation Reply: Accepted or Bye Command will follow.

The Host should record the IP and Port the Client used to send the request. This makes up the identifier to know which Client is attempting to create a Session. All data from the Host to the Client must use this Port and IP.

If a Host receives an Invitation from a remote Client with which it is already in a Session, then it shall respond with *Invitation Reply: Accepted*. See Section ***6.5***.

If a Client (without Host functionality on the given port) receives an Invitation Command, then it shall respond with *NAK* with reason 0x02 (Command not Expected). See Section ***6.15***.

Table 10 Invitation Command

Field	Size (bytes)	Values	Description
Command Code	1	0x01	
Command Payload Length (pl)	1	2...36	Payload length in 32-bit words
Command Specific Data 1 (csd1)	1	1…25	Length in 32-bit words of the UMP Endpoint Name.
Command Specific Data 2	1	bitmap	Capabilities
UMP Endpoint Name	(csd1*4)	String	The UMP Endpoint Name of the Client (UTF-8, max length 98 bytes)
Product Instance Id	(pl-csd1)*4	String	The Product Instance Id of this Client (ASCII, max length 42 bytes)

Capabilities

The Capabilities field is a one-byte bitmap:

Figure 8 Bitmap Format for Capabilities

Table 11 Capabilities for Invitation

Flag	Description
D0	Client supports sending Invitation with Authentication
D1	Client supports sending Invitation with User Authentication
D2…D7	Reserved

If the Client sets bit D0, then Client supports sending the Invitation with Authentication Command. The Host may ask the Client to use password authentication (see Section ***6.7 Invitation Reply: Authentication Required***).

If the Client sets bit D1, then Client supports sending Invitation with User Authentication. The Host may ask the Client to use user authentication (see Section ***6.8 Invitation Reply: User Authentication Required***).

If the Client does not support the Invitation mechanism preferred by the Host, then the Host may use a fallback, for example by asking the user for manual confirmation of the Session.

If the Host does not support Pending Invitation fallback and requires Authentication, and the Client does not support a matching Authentication method, then the Host shall send a Bye Command with reason 0x45 (No Matching Authentication Method).

UMP Endpoint Name

The Client shall provide a name to the Host. This name shall be the same as reported by an Endpoint Name Notification message. The UMP Endpoint Name field is a variable length UTF-8 string of up to 98 bytes. For more information on string encoding, see Section ***5.3.***

Product Instance Id

The Client shall provide a Product Instance Id to the Host. This Product Instance Id reported here shall be the same as reported by a Product Instance Id Notification message. See the UMP and MIDI 2.0 Protocol specification ***[MA06]***.

The Product Instance Id is a variable length ASCII string of up to 42 bytes consisting of the ASCII ordinals 32-126. See Section ***5.3*** for string encoding.

Example Command Packet

For an example of a complete UDP packet containing an Invitation Command, see Appendix ***A.1***.

6.5 Invitation Reply: Accepted

The Host may send an Invitation Reply: Accepted Command to the Client to accept a pending Session. This will start the Session.

Subsequently, UMP Command Packets may be sent by either side.

If a Client receives this Command when it is already in an Established Session with the Host, then it shall ignore it.

If a Client receives this Command when it is not in a Pending Session with the Host, then the Client shall send a Bye Command to the Host with reason 0x06 (No Pending Invitation).

Table 12 Invitation Reply: Accepted Command

Field	Size (bytes)	Values	Description
Command Code	1	0x10	
Command Payload Length (pl)	1	2..36	(in 32-bit words)
Command Specific Data 1 (csd1)	1	1…25	Length in 32-bit words of the UMP Endpoint Name.
Command Specific Data 2	1	0	Reserved
UMP Endpoint Name	(csd1*4)	string	The UMP Endpoint Name of the Host (UTF-8, max length 98 bytes)

Product Instance Id	(pl-csd1)*4	String	The Product Instance Id of this Host (ASCII, max length 42 bytes)

UMP Endpoint Name

The Host shall provide a name to the Client. This shall be the same as the UMP Endpoint Name as reported by an Endpoint Name Notification message. See the UMP and MIDI 2.0 Protocol specification ***[MA06]***.

If the Host is exposed via mDNS, it shall be the same name as provided in the UMPEndpointName TXT record. The name does not need to match the DNS host name. See the UMP Endpoint Name field of the Invitation Command for more information (Section ***6.4 Invitation***).

Product Instance Id

The Host shall provide a Product Instance Id to the Client. This is the same as the UMP Product Instance Id as reported by a Product Instance Id Notification message. See the UMP and MIDI 2.0 Protocol specification ***[MA06]***.

If the Host is exposed via mDNS, it shall be the same name as provided in the ProductInstanceId TXT record. See the Product Instance Id field of the Invitation Command for more information (Section ***6.4 Invitation***).

6.6 Invitation Reply: Pending

A Host may send an Invitation Reply Pending Command to inform the Client that the Host needs some time to determine if it can accept the invitation, or under which conditions it can do so (authentication). The Client shall wait for a subsequent reply from the Host.

> For example, this might occur when the Host asks the user for permission and is waiting for user acknowledgment.

If a Client receives this Command when not in a Pending Invitation or Established Session state, then the Client shall send a Bye Command to the Host with reason 0x06 (No Pending Invitation).

If a Host receives this Command, then the Host shall respond with NAK, reason 0x02 (Command Not Expected). See Section ***6.15***.

Table 13 Invitation Reply Pending Command

Field	Size (bytes)	Values	Description
Command Code	1	0x11	
Command Payload Length (pl)	1	2..36	
Command Specific Data 1 (csd1)	1	1…25	Length in 32-bit words of the UMP Endpoint Name.
Command Specific Data 2	1	0	Reserved
UMP Endpoint Name	(csd1*4)	string	The UMP Endpoint Name of the Host (UTF-8, max length 98 bytes)
Product Instance Id	(pl-csd1)*4	String	The Product Instance Id of this Host (ASCII, max length 42 bytes)

6.7 Invitation Reply: Authentication Required

A Host may send an Invitation Reply: Authentication Required Command to the Client when the Host cannot accept the Invitation because a form of authentication is required.

The Invitation Reply: Authentication Required Command shall only be sent if the Client set the "Client supports sending Invitation with Authentication" flag in the Capabilities field of the Invitation Command.

The Command includes a random cryptographic nonce (CryptoNonce) which the Client shall use for a subsequent Invitation with Authentication Command.

The Host shall remember the CryptoNonce for the remote Client, so that the Host can use the CryptoNonce to verify the expected Invitation with Authentication Command. Upon receiving duplicate Invitation Commands from the same Client, the Host should reply with the same CryptoNonce. Every new Session shall use a new CryptoNonce, even for the same Client.

The original Invitation is terminated with this Command.

A Host may also send this Command in response to an Invitation with Authentication when the provided username and Authentication Digest are not correct. In that case, the Host should delay sending the reply so that a brute force attack will not be practical. A good approach is to start with a short delay (e.g., 100ms) and double it for every subsequent failed authentication.

If a Host receives this Command, then the Host shall respond with NAK, with reason 0x02 (Command Not Expected). See Section ***6.15***.

If a Client receives this Command without having sent an Invitation Command, then the Client shall respond with NAK, with reason 0x02 (Command Not Expected). See Section ***6.15***.

Table 14 Invitation Reply: Authentication Required Command

Field	Size (bytes)	Values	Description
Command Code	1	0x12	
Command Payload Length (pl)	2	6…40	
Command Specific Data 1 (csd1)	1	1..25	Length in 32-bit words of the UMP Endpoint Name field.
Command Specific Data 2	1	state	Authentication State
CryptoNonce	16	ASCII	A cryptographic nonce (random value) to be used for authentication
UMP Endpoint Name	(csd1*4) - 16	string	The UMP Endpoint Name of the Host (UTF-8, max length 98 bytes)
Product Instance Id	(pl-csd1)*4 - 16	String	The Product Instance Id of this Host (ASCII, max length 42 bytes)

Table 15 Values for Authentication State

Value	Description
0x00	First authentication request

0x01	Previously provided Authentication Digest is not correct
0x02…0xFF	Reserved

Authentication State

If the Host sends an Invitation Reply: Authentication Required Command in response to an Invitation Command, then it shall set the Authentication State field to 0x00 (First authentication request).

If the Host receives an Invitation with Authentication Command and it fails to authenticate the Invitation, then it shall reply with either of the two following Commands:

- Send an Invitation Reply: Authentication Required Command with the Authentication State field set to 0x01 (Previously provided Authentication Digest is not correct).
- Send a Bye Command, with the Bye Reason field set to 0x43 (Authentication failed) to terminate the authentication.

6.8 Invitation Reply: User Authentication Required

The Host may send an Invitation Reply: User Authentication Required Command to the Client when it cannot accept the Invitation because a form of user authentication is required.

The Invitation Reply: User Authentication Required Command shall only be sent if the Client set the “Client supports User Authentication” flag in the Capabilities field of the Invitation Command.

The Invitation Reply: User Authentication Required includes a CryptoNonce which the Client shall use for a subsequent Invitation with authentication.

As in Section ***6.7***, the Host shall remember the CryptoNonce for the remote Client so that the Host can use the CryptoNonce to verify the expected Invitation with Authentication Command. Upon receiving duplicate Invitation Commands from the same Client, the Host should reply with the same CryptoNonce. Every new Session requires a new CryptoNonce, even for the same Client.

The original Invitation is terminated with this Command.

This Host also sends this Command in response to an Invitation with User Authentication when the provided username and Authentication Digest are not correct. In that case, the Host should delay sending the reply so that a brute force attack will not be practical. A good approach is to start with a short delay (e.g., 100ms) and double it for every subsequent failed authentication.

If a Host receives this Command, then the Host shall respond with NAK, with reason 0x02 (Command Not Expected). See Section ***6.15***.

If a Client receives this Command without having sent an Invitation Command, then the Client shall respond with NAK, with reason 0x02 (Command not Expected). See Section ***6.15***.

Table 16 Invitation Reply: User Authentication Required Command

Field	Size (bytes)	Values	Description
Command Code	1	0x13	
Command Payload Length (pl)	2	6…29	
Command Specific Data 1 (csd1)	1	1..25	Length, in 32-bit words, of the UMP Endpoint Name.

Command Specific Data 2	1	state	Authentication State
CryptoNonce	16	ASCII	A cryptographic nonce (random value) to be used for authentication
UMP Endpoint Name	(csd1*4) - 16	string	The UMP Endpoint Name of the Host (UTF-8, max length 98 bytes)
Product Instance Id	(pl-csd1-4)*4	String	The Product Instance Id of this Host (ASCII, max length 42 bytes)

Table 17 Values for Invitation Reply: User Authentication Required

Value	Description
0x00	First authentication request
0x01	Previously provided Authentication Digest is not correct
0x02	Previously provided username is not found
0x03…0xFF	Reserved

Authentication State

If the Host sends an Invitation Reply: User Authentication Required Command in response to an Invitation Command, then it shall set the Authentication State field to 0x00 (First authentication request).

If the Host receives an Invitation with Authentication Command and it fails to authenticate the Invitation, then it shall reply with either of the two following Commands:

- Send an Invitation Reply: Authentication Required Command with the Authentication State field set to 0x01 (Previously provided Authentication Digest is not correct) or 0x02 (Previously provided username is not found).
- Send a Bye Command, with the Bye Reason field set to 0x43 (Authentication failed) to terminate the authentication.

6.9 Invitation with Authentication

A Client should send an Invitation with Authentication Command after receiving an Invitation Reply: Authentication Required Command from the Host. This Command uses the received CryptoNonce and a shared secret to form the Authentication Digest.

The shared secret is typically a 4- or 6-digit number (encoded using ASCII), or an alpha-numeric password, which is known by the Host. The Client should prompt the user to enter the secret, or use a cached version.

When the Host receives this Command, the Host calculates the Authentication Digest using the CryptoNonce (as previously sent to the Client), and the shared secret. Then the Host compares the Authentication Digest with the one provided by the Client. If they are the same, the Host shall respond with Invitation Reply: Accepted to start the session. If the Authentication Digest does not match, the Host shall respond with an Invitation Reply: Authentication Required Command, indicating in the Flags field that authentication failed.

If the Host receives an Invitation with Authentication Command but has not received a prior Invitation Command, then the Host should terminate this Invitation with a Bye Command with reason 0x41 (Rejected because no prior invitation received).

After an Invitation with Authentication Command is sent, the Invitation is pending until a Bye Command or Invitation Reply is received. Either side can terminate a pending Invitation with a Bye Command.

The Client should repeat this Command until a reply is received. After a reasonable time without a reply, the Client may consider the Invitation failed and should terminate it with a Bye Command.

If the Host receives an Invitation with Authentication Command but has not received a prior Invitation Command, then the Host should terminate this Invitation with a Bye Command with reason 0x41 (Rejected because no prior invitation received).

Creating the Authentication Digest

The Client and Host shall generate the SHA-256 Authentication Digest by concatenating the CryptoNonce and the shared secret into one string which is hashed using SHA-256. This can be illustrated like this:

```
hash = to_sha256_digest("<CryptoNonce><Shared Secret>")
```

For example, assume the following:

CryptoNonce:	`nUWrn*@#$hjfwnkL`
Shared Secret:	`5483`

Now the string to hash using SHA-256 is:

```
nUWrn*@#$hjfwnkL5483
```

This results in this 32-byte Authentication Digest:

```
0x67 0x6E 0xBE 0x82 0x58 0x7C 0xEC 0xA8
0xF8 0x2F 0xC3 0x33 0xD7 0x87 0x95 0x1E
0xEC 0x2B 0x00 0xAD 0x31 0x61 0x3C 0xC4
0xDB 0x18 0xCF 0x27 0x37 0x3A 0xFB 0x82
```

Table 18 Invitation with Authentication Command

Field	Size (bytes)	Values	Description
Command Code	1	0x02	
Command Payload Length (pl)	1	8	
Command Specific Data	2	0	Reserved
Auth Digest	32	SHA-256 Digest	Binary SHA-256 Authentication Digest to be used to verify authentication

6.10 Invitation with User Authentication

If a Client supports user authentication, then the Client should send an Invitation with User Authentication Command after receiving an Invitation Reply: User Authentication Required Command from the Host. The Client uses the received CryptoNonce, a username, and a user password to form the Authentication Digest.

The Host should have stored a list of permitted users and their passwords. The Client may store one or more user/password pairs, or may allow the user to enter them.

When the Host receives this Command, the Host must consult its user list to find the given username and its password. Then the Host shall proceed with either of the 2 following procedures:

1. If the Host finds the given username, the Host calculates the Authentication Digest using the CryptoNonce (as previously sent to the Client), the provided username, and the password it has stored for this user. Then the Host compares the Authentication Digest with the one provided by the Client:

   a. If the Client’s Authentication Digest and the Host’s Authentication Digest are the same, then the Host responds with Invitation Reply: Accepted Command to accept the Session.
   b. If the Client’s Authentication Digest and the Host’s Authentication Digest are not the same, then the Host responds with an Invitation Reply: User Authentication Required Command, indicating in the Authentication State field that authentication failed.
2. If the Host cannot find the given username, then the Host shall respond with an Invitation Reply: User Authentication Required message, indicating in the Flags field that the user was not found.

After an Invitation with User Authentication Command is sent, the Invitation is pending until a Bye or Invitation Reply is received. Either side may terminate a pending Invitation with a Bye Command.

The Client should repeat this Command until a reply is received. After a reasonable time without a reply, the Client may consider the Invitation failed and should terminate it with a Bye Command.

If the Host receives an Invitation with User Authentication Command but has not received a prior Invitation Command, then the Host should terminate this Invitation with a Bye Command with reason 0x41 (Rejected because no prior invitation received).

Creating the Authentication Digest

The Client and Host shall generate the SHA-256 Authentication Digest by concatenating the CryptoNonce, the username, and the password into one string which is hashed using SHA-256. This can be illustrated like this:

```
hash = to_sha256_digest("<CryptoNonce><Username><Password>")
```

For example, assume the following:

CryptoNonce:	`XI|~=NNRVaD;XCPL`
Username:	`Rosa`
Password:	`RPBqBno`

Now the string to hash using SHA-256 is:

```
XI|~=NNRVaD;XCPLRosaRPBqBno
```

This results in this 32-byte Authentication Digest:

```
0x3A 0x2D 0x5E 0xBE 0xEF 0x92 0xE8 0x46
0x35 0x35 0xC2 0x7F 0xB4 0xB7 0xB3 0xE2
0xD1 0xAC 0xF5 0x7A 0x84 0x4C 0x6D 0x08
0x08 0xE0 0x41 0xC1 0x02 0xB7 0xFF 0x1A
```

Table 19 Invitation with User Authentication Command

Field	Size (bytes)	Values	Description
Command Code	1	0x03	
Command Payload Length (pl)	1	8..255	
Command Specific Data	2	0	Reserved
Auth Digest	32	SHA-256 Digest	Binary SHA-256 Authentication Digest to be used to verify authentication
Username	(pl*4) - 32	string	The username (UTF-8)

6.11 Session Reset

A Session Reset Command is used to reset the Sequence Number to zero. Reasons to use this command might include:

- When packets have been lost, 2 devices are out of sync, data recovery is not possible (such as a Retransmit is not available).
- If a Device is unable to cope with incoming messages, such as a large System Exclusive message and wishes to abort.

A Session Reset Command may be sent by either Host or Client at any time when in an active Session. The Session Reset Command should be repeated until a Session Reset Reply Command is received or a timeout occurs. While waiting for the Session Reset Reply command, the Device shall not send any UMP Data Commands, and may ignore incoming UMP Data Commands until it receives a Session Reset Reply Command and UMP Data Command with Sequence Number 0. On reception of the Session Reset Reply Command, the Session shall be reset. On timeout, the session shall be terminated by sending the Bye Command with reason 0x04 (Timeout).

The Session Reset Command should be repeated until a Session Reset Reply Command is received or a timeout occurs. On timeout, the session shall be terminated by sending the Bye Command with reason 0x04 (Timeout).

On reception of the Session Reset Command, the Receiver shall respond with the Session Reset Reply Command and reset the session.

If the receiver of a Session Reset Command is not in an active Session with the Sender, it shall send a Bye Command with reason 0x05 (Session Not Established, see Section 6.16).

How to Reset the Session

Both devices shall act as if the Session had been torn down and initiated.:

- Sequence Numbers shall be set to 0
- Buffers used for FEC and Retransmit shall be flushed
- Allow sending UMP Data again

Both Devices might perform other actions to reset their own state such as notifying the application layer, so it can reset state such as stop hanging notes, for example by issuing an All Notes Off.

Table 20 Session Reset Command

Field	Size (bytes)	Values	Description
Command Code	1	0x82	
Command Payload Length (pl)	1	0	
Command Specific Data	2	0	Reserved

6.12 Session Reset Reply

Upon reception of a Session Reset Command, the receiver shall respond with the Session Reset Reply Command.

On reception of the Session Reset Reply Command, the receiver shall reset the Session, as outlined in the description of the Session Reset Command.

If the receiver of a Session Reset Reply Command has not sent a prior Session Reset Command, then the receiver should reset the Session by sending a Session Reset Command.

If the receiver is not in an active Session with the sender, it shall send a Bye Command with reason 0x05 (Session Not Established, see Section 6.16).

Table 21 Session Reset Reply Command

Field	Size (bytes)	Values	Description
Command Code	1	0x83	
Command Payload Length (pl)	1	0	
Command Specific Data	2	0	Reserved

6.13 Ping

A Host or Client may send a Ping Command at any time and in any Session State.

The Ping Command is used in the following manner:

- Host or Client A sets the Ping Id field to an arbitrary number and then sends the Ping Command to B.
- On reception, B replies with the Ping Reply Command (see Section *6.14*) with a copy of the Ping Id from the Ping Command received from A.

Regularly exchanging Ping Commands might keep firewalls open and allows detection of a stale remote entity, using a timeout. If multiple Ping Commands are not replied to for an extended amount of time, the Session should be considered terminated. In this case, the Host or Client should send a Bye Command with reason 0x04 (Timeout, see Section 6.16).

Table 22 Ping Command

Field	Size (bytes)	Values	Description
Command Code	1	0x20	
Command Payload Length (pl)	1	1	
Command Specific Data	2	0	Reserved
Ping Id	4	32-bit unsigned	Ping Id, chosen by sender

Ping Id

The Sender may place any value into the Ping Id field of the Command as an outgoing identifier for that Ping Command. Ping Id shall not be used for any purpose other than this identifier. The purpose of the Ping Id field is to match a received Ping Reply Command to a previously sent Ping Command.

6.14 Ping Reply

Upon reception of a Ping Command, the Receiver shall respond with the Ping Reply Command as soon as possible. If possible, any other pending Commands should be bypassed so that latency measurement using the Ping round-trip time is not affected by other Commands.

If a Client or Host receives a Ping Reply without having sent a prior Ping Command, or the Ping Id does not match the previously sent Ping Id in a Ping Command, it may send a NAK Command with reason 0x20 (Bad Ping Reply, see Section 6.16).

Table 23 Ping Reply Command

Field	Size (bytes)	Values	Description

Command Code	1	0x21	
Command Payload Length (pl)	1	1	
Command Specific Data	2	0	Reserved
Ping Id	4	Ping Id	The Ping Id copied from the received Ping Command

6.15 NAK

A Host or Client sends a NAK Command to report a problem with a received Command. A pending Session or an Established Session will not be terminated because of a NAK Command.

Table 24 NAK Command

Field	Size (bytes)	Values	Description
Command Code	1	0x8F	
Command Payload Length (pl)	1	1..255	
Command Specific Data 1	1	Enum	NAK Reason: A code specifying the reason for this NAK Command.
Command Specific Data 2	1	0	Reserved
NAK'ed Command Header	4		Header (first word) of the original Command, copied directly from the received Command.
Text Message	(pl - 1) * 4	string	An optional additional text describing the reason, suitable for presentation to the user (UTF-8)

NAK Reason

The NAK Reason specifies the reason for replying with NAK.

Table 25 List of NAK Reasons

NAK Reason Value	Reason	Description
0x00	Other	The Text Message field provides the NAK Reason
0x01	Command Not Supported	An endpoint uses this NAK Reason whenever it receives a Command that it does not support.
0x02	Command Not Expected	An endpoint uses this NAK Reason if it normally supports the received Command, but it was not expected at this time. For example, an unsolicited Ping Reply.
0x03	Command Malformed	A receiver uses this NAK reason when a Command does not include the required payload, or uses incorrect values that prevent

		parsing or using the respective Command. Do not send it if the payload is longer than expected or if Reserved fields or bits contain data.
0x20	Bad Ping Reply	An Endpoint uses this NAK code if it receives a Ping Reply which does not contain the expected Ping Id.

Text Message

An optional text providing additional information for presentation to the user. The text should be complementary to the Reason (which may also be displayed to the user), not a repeat of the description of the NAK Reason code. The text message shall be UTF-8 encoded. If the text ends in the middle of a word, then the remaining data bytes shall be set to 0x00 to indicate the end of the text.

6.16 Bye

A Host or Client shall send a Bye Command to end a Session or to terminate a pending Invitation. The Bye Command should be sent repeatedly until a Bye Reply Command is received, or until a timeout occurs.

If a Device receives a Bye Command, the Device shall send the Bye Reply Command back to the sender to terminate the Session. Because the Bye Command might be repeated, the Bye Reply shall also be sent if there is no Pending or Established Session. Only send a single Bye Reply in response to each Bye Command.

Table 26 Bye Command

Field	Size (bytes)	Values	Description
Command Code	1	0xF0	
Command Payload Length (pl)	1	0..255	
Command Specific Data 1	1	Enum	Bye Reason: An informative reason code specifying the reason for terminating the session.
Command Specific Data 2	1	0	Reserved
Text Message	pl*4	string	An optional message string. The text message shall be UTF-8 encoded, without a Byte Order Mark. If the text ends in the middle of a word, then the remaining data bytes shall be set to 0x00 to indicate the end of the text.

Bye Reason

The Bye Reason code specifies the cause reason for terminating the session.

Table 27 List of Bye Reasons

Bye Reason Value	Reason	Examples
— Sent by Either Client or Host —		
0x00	Unknown or Undefined	

0x01	User terminated session	
0x02	Power Down	
0x03	Too Many Missing UMP Packets - cannot recover.	
0x04	Timeout	e.g. too many bad/missing ping responses
0x05	Session Not Established	e.g. One Endpoint believes it is in an Established Session, the other is not in an Established Session
0x06	No Pending Session	
0x07	Protocol Error	e.g. UMP Endpoint Name or Product Instance Id missing from Invitation command.
— Sent from Host to Client —		
0x40	Invitation Failed: too many opened sessions	
0x41	Invitation With Authentication Rejected: missing prior invitation attempt without authentication.	
0x42	Invitation Rejected: user did not accept session	
0x43	Invitation Rejected: authentication failed	
0x44	Invitation Rejected: username not found	
0x45	No Matching Authentication Method	
— Sent from Client to Host —		
0x80	Invitation Canceled	

6.17 Bye Reply

A Host or Client shall send a Bye Reply Command as a response to a received Bye Command. The Command confirms that the Device has received the Bye Command and recognizes that the Session is terminated.

If a Device receives a Bye Reply Command, the Device shall stop repeatedly sending the Bye Command. If there is no Established Session with the sender of the Bye Reply Command, the receiver shall ignore the Bye Reply Command.

Table 28 Bye Reply Command

Field	Size (bytes)	Values	Description
Command Code	1	0xF1	
Command Payload Length (pl)	1	0	

Command Specific Data	2	0	Reserved

7 UMP Data Transfer

7.1 UMP Data Command

The UMP Data Command is used to transfer one or more UMP messages.

After a Session has been established and as long as the Session is in Established state, both Client and Host may send UMP Data Commands.

If a Device receives the UMP Data Command when not in an Established Session, then it shall respond with a Bye Command, reason 0x05 (Session not Established).

Table 29 UMP Data Command

Field	Size (bytes)	Values	Description
Command Code	1	0xFF	
Command Payload Length (pl)	1	0…64	Number of 32-bit words
Command Specific Data	2	16-bit unsigned	Sequence Number
UMP Data	pl*4	UMP	Zero or more complete UMP messages

Command Payload Length

This declares the length of the payload in 32-bit words. The length shall not exceed 64 words.

Sequence Number

This is the Sequence Number for this Client or Host for the specific Session. A Sender shall use Sequence Number 0x0000 for the first UMP Data Command and shall increment the Sequence Number for every following UMP Data Command sent. Sequence Numbers wrap around after 0xFFFF, back to 0x0000. If the UMP Data payload contains multiple UMP messages, the Sequence Number applies to all of them. On reception, the Receiver should verify the Sequence Number to detect missing, duplicate, or out of order UMP Data Commands. See Section *7.2.2* for more information about implementing Forward Error Correction using the Sequence Number.

UMP Data

The payload section of the UMP Data Command may be either of the following:

- One or more UMPs
- A zero length payload. See Section *7.2.1*.

If a single UMP Data Command contains multiple UMPs, then the Sender and Receiver shall preserve the original sequential order of the UMP. The messages may be rendered simultaneously.

Each UMP packet shall be transmitted in whole within a UMP Data Command, not split across multiple UMP Data Commands.

A single UDP payload may contain multiple UMP Data Commands. This is required for implementing Forward Error Correction (see Section *7.2.2*), and can also be used for sending more simultaneous UMP messages than fit into a single UMP Data Command.

For examples of complete UDP packets containing UMP Data Commands, see Appendix *A.1*.

7.2 Mechanisms for Protecting Data Stream Integrity

General Considerations

Because the UDP / IP stack uses checksums, UDP packets are assumed to arrive in unaltered form. Consequently, the most common data stream integrity problems that Network MIDI 2.0 (UDP) implementations will need to handle are missing, duplicate, and out-of-order UDP packets.

Most Session Commands are sent repeatedly until a reply is received. Typically, Session Commands will eventually reach the destination, even if some packets are lost. Refer to the Command descriptions for details which ones to repeat.

Some Session Commands such as NAK do not have an associated reply. Such Commands are only sent once.

UMP Data Commands use a sequence number to declare and maintain the sequential order of UMP packets and for data integrity mechanisms.

There are two different mechanisms for recovering from missing UMP Data Commands: Forward Error Correction (FEC) described in Section *7.2.2* and Retransmit described in Section *7.2.3*.

Ignoring Duplicate UMP Data Commands

The underlying transport might cause duplication of UDP packets. Also the FEC mechanism uses duplicate UMP Data Commands to create redundancy. Therefore, Receivers ignore UMP Data Commands with a Sequence Number which has already been received and processed.

Handling Out-Of-Order UMP Data Commands

The underlying transport might cause out of order packets. For example, if the sender sends UMP Data with sequence number 0x0001 and then followed by a UMP Data Command with sequence number 0x0002, these packets may arrive in reversed order, so that 0x0002 is received before 0x0001. The Sequence Number, Forward Error Correction, and other mechanisms in this specification will mitigate and assist with handling unordered UMP Data Commands.

Detecting Missing UMP Data Commands

A Receiver remembers the Sequence Number of the last received UMP Data Command packet. A missing packet can be detected by comparing the expected Sequence Number with the received Sequence Number.

Zero Length UMP Data Commands

UMP Data Commands of zero length are used to help maintain data integrity and to help indicate that a session is still active when a Sender has a period where there is no UMP data to send. See Section *7.2.1*.

Recovery Mechanisms

This specification defines two different mechanisms to recover from missing UMP Data Commands: Forward Error Correction, and the Retransmit Request Command.

Forward Error Correction (FEC)

With Forward Error Correction, the Sender repeats previous UMP Data Commands when sending a new UMP Data Command, so that the Receiver can seamlessly recover from a missing packet. It is recommended that all Senders implement FEC. Receivers do not need special handling to implement FEC; FEC packets will appear as duplicate UMP Data Commands, which every Receiver must handle anyway.

See Section *7.2.2* for the definition and rules for the use of FEC.

Retransmit

The Retransmit Request Command (Section 7.2.3) allows a Receiver to re-request missing UMP Data Commands. Retransmit functionality is optional but recommended for implementation by all Hosts and Clients.

7.2.1 Declaring Idle Period: Zero Length UMP Data Command

If a Sender has a period where there is no UMP data to send, the Sender shall send a Zero Length UMP Data Command to inform the Receiver that the Sender currently has no further UMP data.

That first Zero Length UMP Data Command shall be sent within 300ms after the most recent UMP Data Command which had a non-zero length. However, a shorter interval period should be used for timing-critical applications.

If the Sender continues to have no data to send, then the Sender should send subsequent Zero Length UMP Data Commands with increasingly longer interval times. The interval times for such further Zero Length UMP Data Commands may be determined by the Sender. The Sender should expand the interval between each further Zero Length UMP Data Command and eventually stop sending Zero Length UMP Data Commands.

Zero Length Data Commands shall use Sequence Numbers in the same manner as any other UMP Data Command. See Section ***5.6***.

The Sender should consider that the Receiver may have restrictions such as battery operation or limited processing in which it would prefer to not consistently receive data.

7.2.2 Forward Error Correction (FEC)

Every Client and Host should implement FEC when sending UMP packets by including two previously sent UMP Data Commands. A Sender may also choose to send only a single FEC repeat, or more than two FEC repeats. However, research has shown that using FEC with more than two repeats does not significantly improve data integrity ***[EIA01]***.

Every Device receiving a UDP packet with UMP data shall be able to skip previously received UMP Data Commands. FEC can be used with or without Retransmit functionality.

See below for an example where two previous UDP Data Commands are sent before a new UMP Data Command.

If FEC is implemented and the Sender enters an idle period (see Section 7.2.1), the Sender should send multiple UDP packets with the last UMP Data Commands prior to the idle period with Zero Length UMP Data Command(s). Such UDP packets should be included up to the number of FEC data repeats the Sender is currently using for delivery of the prior UMP Data Commands.

> For example, if this application determines that the idle period is 10ms and is sending two previous UMP Data Commands for FEC, then after the first Zero Data UMP Packet, up to two further Zero Data UMP Packets should be sent at 10ms intervals if no further UMP data is to be sent. After the FEC mechanism has been delivered, subsequent Zero Length UMP Data might be sent with longer interval times (see Section 7.2.1).

FEC Packet Order

Previous UMP payloads shall be prepended in the order in which they were sent. This is to allow the receiving Device to read each UMP Data Command in order in which it is received and just skip over the UMP Data Commands it has already processed.

FEC Example

The following example shows a UDP packet with one new UMP Command with Sequence Number 0x0023, and two included FEC repeats of previously sent UMP Commands with Sequence Numbers 0x0021 and 0x0022.

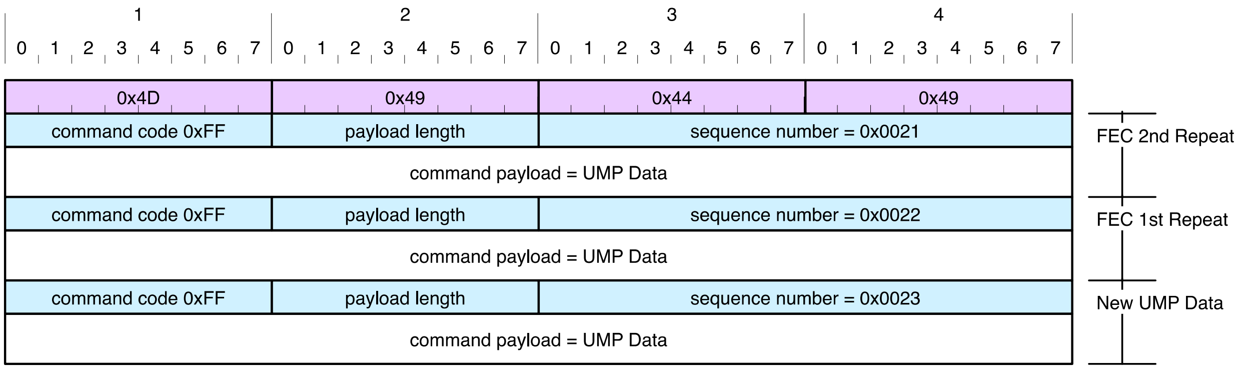

Figure 9 Example for UDP Packet with FEC

For an example of a sequence of UMP Data Commands using FEC, see Appendix ***A.2***.

7.2.3 Retransmit Request Command

A Device may send a Retransmit Request Command to ask the receiving entity to resend UMP Data Command(s). One or more requested UMP Data Commands are identified by Sequence Number in the Retransmit Request Command.

The Device should delay sending the Retransmit Request Command for a short duration, for example 10 milliseconds. That will help recovering from out of order packets and it prevents sending Retransmit Requests too often.

The Retransmit Request Command should be repeated (with increasing delay) until the Device receives the requested UMP Data Command(s), a Retransmit Error Command, a NAK Command, or until a timeout occurs.

When a Device receives the Retransmit Command, it has these options:

- If the Device can retransmit the requested packets, it will do so by sending one or more UDP packets with the requested Command Packets.
- If the Device does not have the requested packets available for retransmit anymore, it replies with a Retransmit Error Command Packet.
- If the Device does not support Retransmit functionality at all, it replies with NAK and appropriate reason code.

When a Device receives a Retransmit Request Command, the Device should retransmit the requested UMP Data Commands starting with the Sequence Number given in the Retransmit Request Command. If the UMP Data Commands for one or more Sequence Numbers are not available in the retransmit buffer for retransmission, then the Device should send a Retransmit Error Command specifying the first missing Sequence Number (see Section ***7.2.4***). All UMP Data Commands that are available in the retransmit buffer following the missing packets should still be retransmitted. The sending entity should always send all requested UMP Data Commands without any gaps.

Note that due to wrap-around of the Sequence Number, the UMP Data Commands to be retransmitted could start with a high number close to the maximum, and then wrap around to 0x0000. The Sender should order the Retransmit buffer considering this wrap-around.

If a Device has already received a request for a Sequence number and has not sent the UMP Data Commands, then it may ignore the Retransmit Request.

If a Device does not implement the Retransmit mechanism, it shall reply to the Retransmit Request Command with a NAK Command with reason 0x01 (Command not supported). See Section ***6.15***. The remote Device should not send Retransmit Request Commands after that.

If a Device receives the Retransmit Request Command outside of an Established Session, then it shall respond with a Bye Command with reason 0x05 (Session Not Established).

The Sender may ignore the Number of UMP Commands field and send all packets since the specified Sequence Number.

Table 30 Retransmit Request Command

Field	Size (bytes)	Values	Description
Command Code	1	0x80	
Command Payload Length (pl)	1	1	
Command Specific Data	2	16-bit unsigned	Sequence Number: The Sequence Number of the first UMP Data Command to be retransmitted
Number of UMP Commands	2	0	Number of UMP Data Commands to retransmit 0x0000 = Send all previously sent UMP Data Commands starting from Sequence Number
Reserved	2	0	

7.2.4 Retransmit Error Command

If a Device which supports the Retransmit mechanism receives a Retransmit Request and is unable to retransmit the requested UMP Data Command(s), then the Device shall send a Retransmit Error Command. A Retransmit Error does not automatically end the Session. The Device shall not retransmit other available UMP Data Commands.

If a Device receives a Retransmit Error Command, the requested UMP data is probably lost and cannot be retrieved. The Device may determine a recovery process appropriate to its own implementation and the unique circumstances, such as triggering an all-notes off.

If a Device receives the Retransmit Error Command outside of an Established Session, then it shall respond with a Bye Command, reason 0x05 (Session Not Established).

Table 31 Retransmit Error Command

Field	Size (bytes)	Values	Description
Command Code	1	0x81	
Command Payload Length (pl)	1	1	
Command Specific Data 1	1	Enum	Error Reason: a code indicating the cause of the Retransmit Error.
Command Specific Data 2	1	0	Reserved
Sequence Number	2	16-bit unsigned	Sequence Number: The Sequence Number of the first UMP Data Command that could be retransmitted
Reserved	2	0	Reserved

Error Reason

The Error Reason code specifies the cause why the sender cannot retransmit one or more requested UMP Data Commands.

Table 32 List of Error Reasons

Error Reason Value	Description
0x00	Unknown
0x01	Transmit buffer does not contain the UMP Data Command for the requested Sequence Number

8 Devices that Act as Both a Host and a Client

Devices which act as a Host may also choose to connect to other Hosts as a Client. Devices which act as both a Host and as a Client and which represent the same UMP Endpoint shall use the same Endpoint Name and Product Instance Id in both roles for:

- DNS Text (TXT) Record (Section 4.4)
- Invitation Command (Section 6.3)
- Invitation Reply: Accepted (Section 6.4)
- Invitation Reply: Pending (Section 6.5)
- Invitation Reply: Authentication Required (Section 6.6)
- Invitation Reply: User Authentication Required (Section 6.7)

By using the same Endpoint Name and Product Instance Id, other Hosts and Clients can better recognize Devices which may already be connected and which match incoming Invitation requests from Clients to a known Host.

The Client and the Host should each use their own UDP Port.

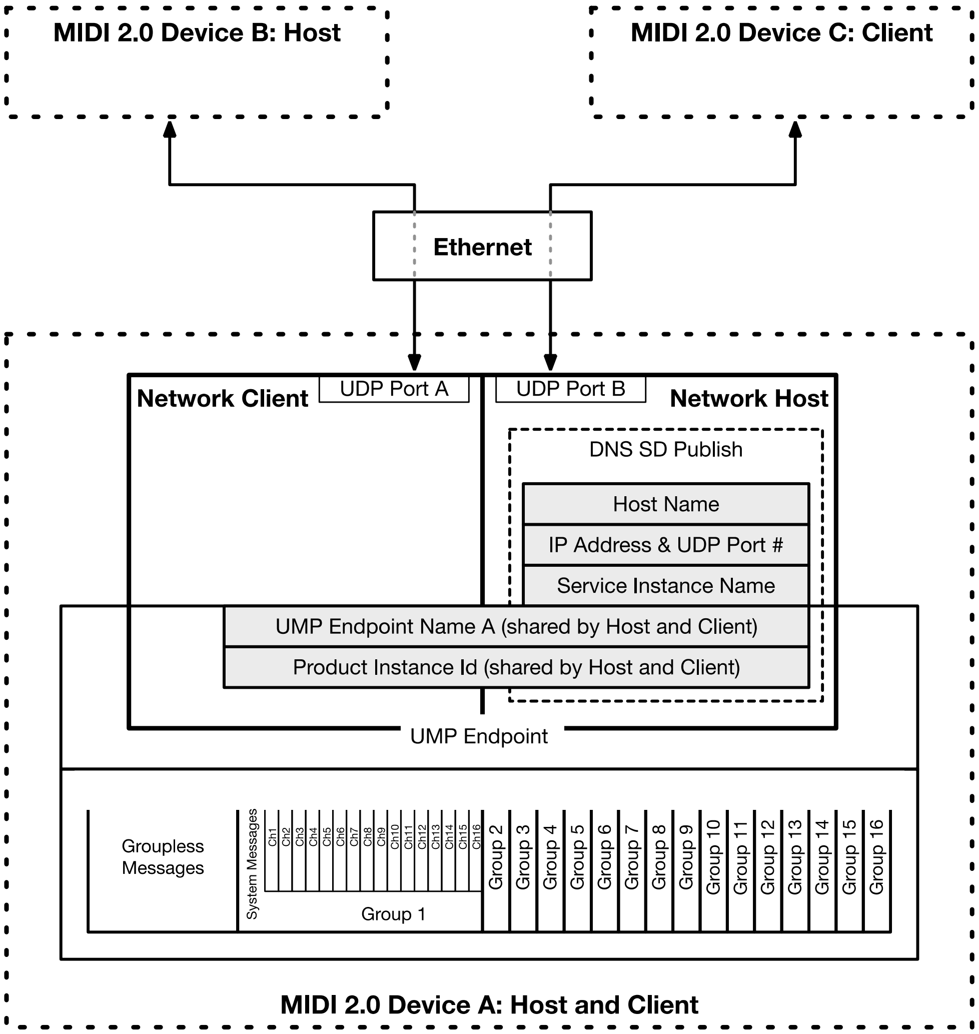

Figure 10: Device Acting as Both Host and Client

9 Device with Multiple UMP Endpoints

A Device may have more than a single UMP Endpoint. Each UMP Endpoint to be exposed on the network shall be represented by an own Host instance with an own UDP port, and an own UMP Endpoint Name. The Product Instance Id and the IP address(es) are usually shared between the Hosts.

Here is a diagram with an example device with two UMP Endpoints which are exposed on the network:

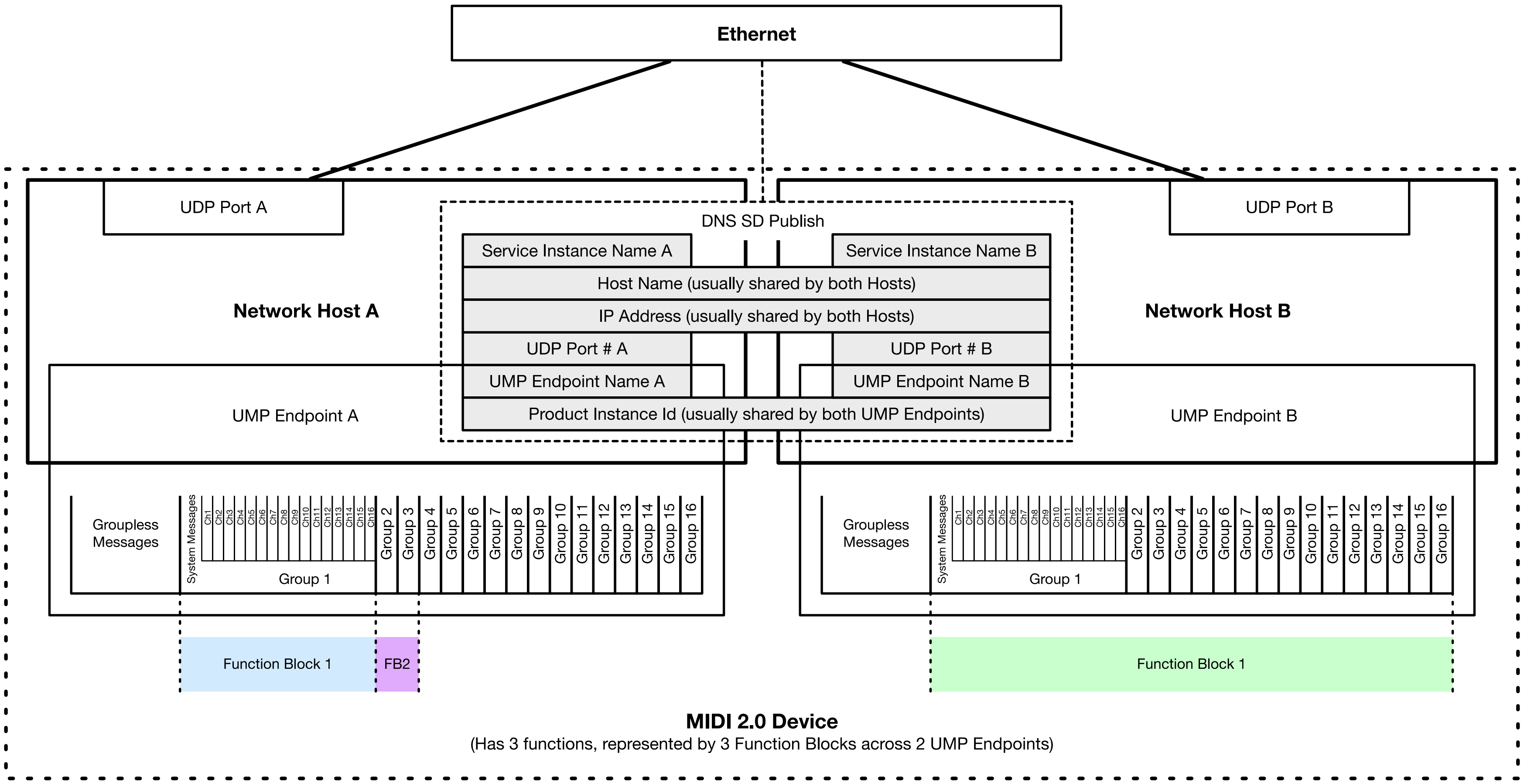
Ethernet
UDP Port A
UDP Port B
DNS SD Publish
Service Instance Name A
Service Instance Name B
Host Name (usually shared by both Hosts)
IP Address (usually shared by both Hosts)
UDP Port # A
UDP Port # B
UMP Endpoint Name A
UMP Endpoint Name B
Product Instance Id (usually shared by both UMP Endpoints)
Network Host A
Network Host B
UMP Endpoint A
UMP Endpoint B
Groupless Messages
System Messages
Ch1 Ch2 Ch3 Ch4 Ch5 Ch6 Ch7 Ch8 Ch9 Ch10 Ch11 Ch12 Ch13 Ch14 Ch15 Ch16
Group 1
Group 2 Group 3 Group 4 Group 5 Group 6 Group 7 Group 8 Group 9 Group 10 Group 11 Group 12 Group 13 Group 14 Group 15 Group 16
Function Block 1
FB2
Function Block 1
MIDI 2.0 Device
(Has 3 functions, represented by 3 Function Blocks across 2 UMP Endpoints)

Figure 11: Device with two UMP Endpoints

Appendix A : Examples

A.1 Examples for UDP Packets

A.1.1 UDP Packet with Invitation Command

Capabilities:	`0x00` (no authentication methods available)
UMP Endpoint Name:	`MyDev`
Product Instance Id:	`8shYe3h5`

0x4D	0x49	0x44	0x49
0x01	0x04	0x02	0x00
'M'	'y'	'D'	'e'
'v'	0x00	0x00	0x00
'8'	's'	'h'	'Y'
'e'	'3'	'h'	'5'

Figure 12 Example UDP Packet with Invitation Command

Note: This example has text for UMP Endpoint Name with 0x00 termination and padding. The Product Instance Id ends on a word boundary, so padding is not used.

A.1.2 UDP Packet with UMP Data

UMP Data:	one UMP message
Seq Number:	16 (0x0010)
UMP Message:	Timing Clock (1 word)
UMP Group:	0 (0x0)

0x4D	0x49	0x44	0x49
0xFF	0x01	0x00	0x10
0x10	0xF8	0x00	0x00

Figure 13 Example UDP Packet with UMP Data

A.1.3 UDP Packet with multiple UMP messages

One UMP Data Command containing two UMP messages:

UMP Data:	two UMP messages
Seq Number:	17 (0x0011)
UMP Message 1:	MIDI 2.0 Note On, Group 6, Channel 1, Note #0x40, attribute type 0, velocity 0x1234.
UMP Message 2:	MIDI 2.0 Note Off, Group 6, Channel 1, Note #0x40, attribute type 0, velocity 0x0100.

0x4D	0x49	0x44	0x49
0xFF	0x04	0x00	0x11
0x45	0x90	0x40	0x00
0x12	0x34	0x00	0x00
0x45	0x80	0x40	0x00
0x01	0x00	0x00	0x00

Figure 14 Example UDP Packet with Multiple UMP Messages

A.1.4 UDP Packet with multiple UMP Data Commands

The same UMP messages as above, but sent in two separate UMP Data Commands.

UMP Data Command 1:

UMP Data:	one UMP messages
Seq Number:	13398 (0x3456)
UMP Message:	MIDI 2.0 Note On, Group 6, Channel 1, Note #0x40, attribute type 0, velocity 0x1234.

UMP Data Command 2:

UMP Data:	one UMP messages
Seq Number:	13399 (0x3457)
UMP Message:	MIDI 2.0 Note Off, Group 6, Channel 1, Note #0x40, attribute type 0, velocity 0x0100.

0x4D	0x49	0x44	0x49
0xFF	0x02	0x34	0x56
0x45	0x90	0x40	0x00
0x12	0x34	0x00	0x00
0xFF	0x02	0x34	0x57
0x45	0x80	0x40	0x00
0x01	0x00	0x00	0x00

Figure 15 Example UDP Packet with Multiple UMP Data Commands

A.2 Example for FEC

The following is an example of UMP payloads being sent from a Client to a Host. The example Client uses up to 2 FEC repeats.

When the Host receives a UDP packet, it reads the UMP Data Command header. If the Sequence Number has already been processed, it can safely skip forward to the next UMP Data Command.

In this example, if UDP Packets #2 and #3 were dropped, then the receiver can safely continue processing UMP Data Commands without any loss of data: it receives the missing UMP Data Commands in the subsequent UDP Packet #4.

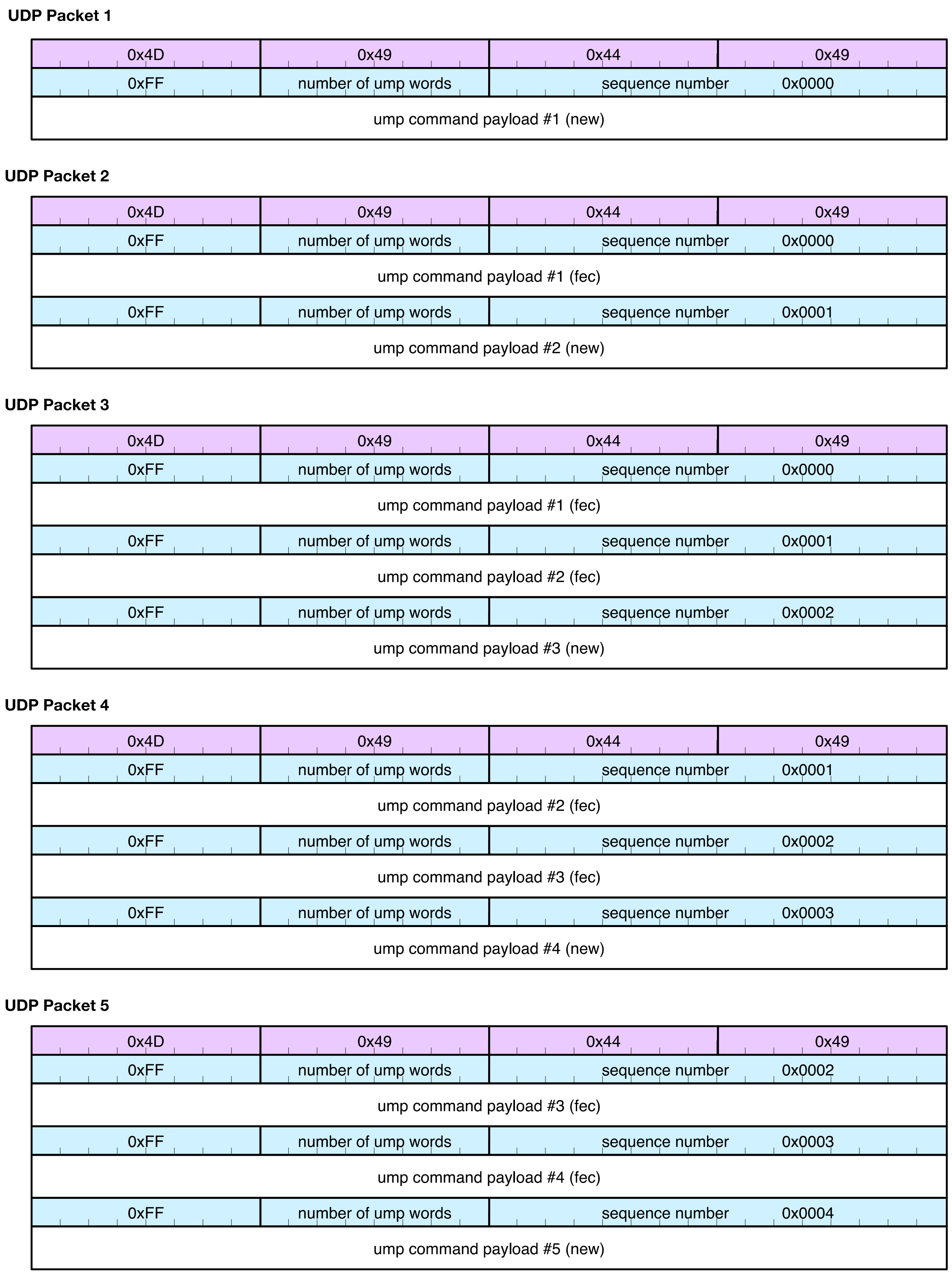

Figure 16 Example Series of UDP Packets with FEC

A.3 Examples for Entering Idle Period With FEC

The following two diagrams show examples of a sequence of UMP Data Commands.

In both examples, the Sender is using FEC with 2 packets and Zero Length UMP Data Commands to signal the beginning of an idle period.

Example 1

UDP Packet Payload - With final UMP Data before period with no more data to send

0x4D	0x49	0x44	0x49
command code 0xFF	payload length	sequence number = 0x0021	
command payload = UMP Data			
command code 0xFF	payload length	sequence number = 0x0022	
command payload = UMP Data			
command code 0xFF	payload length	sequence number = 0x0023	
command payload = UMP Data			

UDP Packet Payload - with FEC repeat for 2 prior UMP Data and Zero length UMP Data

0x4D	0x49	0x44	0x49
command code 0xFF	payload length	sequence number = 0x0022	
command payload = UMP Data			
command code 0xFF	payload length	sequence number = 0x0023	
command payload = UMP Data			
command code 0xFF	payload length =0x00	sequence number = 0x0024	

UDP Packet Payload - with FEC repeat for 1 prior UMP Data and Zero length UMP Data

0x4D	0x49	0x44	0x49
command code 0xFF	payload length	sequence number = 0x0023	
command payload = UMP Data			
command code 0xFF	payload length =0x00	sequence number = 0x0024	

UDP Packet Payload - with Zero length UMP Data

0x4D	0x49	0x44	0x49
command code 0xFF	payload length =0x00	sequence number = 0x0024	

Figure 17: Example 1 UDP Packets with FEC and Zero Length Data Commands

Example 2

UDP Packet Payload - With final UMP Data before period with no more data to send

0x4D	0x49	0x44	0x49
command code 0xFF	payload length	sequence number = 0x0021	
command payload = UMP Data			
command code 0xFF	payload length	sequence number = 0x0022	
command payload = UMP Data			
command code 0xFF	payload length	sequence number = 0x0023	
command payload = UMP Data			

UDP Packet Payload - with FEC repeat for 2 prior UMP Data and Zero length UMP Data

0x4D	0x49	0x44	0x49
command code 0xFF	payload length	sequence number = 0x0022	
command payload = UMP Data			
command code 0xFF	payload length	sequence number = 0x0023	
command payload = UMP Data			
command code 0xFF	payload length =0x00	sequence number = 0x0024	

UDP Packet Payload - with FEC repeat for 1 prior UMP Data and Zero length UMP Data

0x4D	0x49	0x44	0x49
command code 0xFF	payload length	sequence number = 0x0023	
command payload = UMP Data			
command code 0xFF	payload length =0x00	sequence number = 0x0024	
command code 0xFF	payload length =0x00	sequence number = 0x0025	

UDP Packet Payload - with Zero length UMP Data

0x4D	0x49	0x44	0x49
command code 0xFF	payload length =0x00	sequence number = 0x0024	
command code 0xFF	payload length =0x00	sequence number = 0x0025	
command code 0xFF	payload length =0x00	sequence number = 0x0026	

Figure 18: Example 2 UDP Packets with FEC and Zero Length Data Commands

Subsequent UDP packets

Optionally, while no new UMP data is to be sent, in either example above, the Sender may send further UDP packets with Zero Length UMP Data.

http://www.amei.or.jp

https://www.midi.org