MIDI Capability Inquiry (MIDI-CI)

Bidirectional Negotiations for MIDI Devices

Version 1.1
February 20, 2020

Published By:

Association of Musical Electronics Industry AMEI
and
MIDI Manufacturers Association MMA

M2-101-UM

http://www.midi.org

This Version 1.1 of MIDI-CI completely replaces and deprecates MIDI-CI Version 1.0.

PREFACE

MIDI has been a successful tool for more than three decades. The features of MIDI 1.0 continue to work well. The basic semantic language of music does not change and as a result the existing definitions of MIDI as musical control messages continue to work remarkably well.

However, MIDI has not changed to fully take advantage of the new technical environment around it. We want to expand the feature set of MIDI capabilities.

At the same time, we recognize there are several key hurdles and requirements to consider as we make any additions to MIDI:

- Backwards compatibility is a key requirement. Our users expect new MIDI Devices to work seamlessly with MIDI Devices sold over the past 33 years.
- All MIDI Status Bytes are defined. The opcodes and data payloads are defined. It is difficult to define any new messages or change the format of the existing MIDI messages.

Expanding MIDI with new features requires a new protocol with extended MIDI messages. To protect backwards compatibility in an environment with expanded features, Devices need to confirm the capabilities of other connected Devices. When two Devices are connected to each other, they use MIDI 1.0 and confirm each other's capabilities before using new features. If both Devices share support for the same expanded MIDI features, they can agree to use those expanded MIDI features. MIDI-CI provides this mechanism.

MIDI-CI: Solution for Expanding MIDI while Protecting Backwards Compatibility:

MIDI Capability Inquiry (MIDI-CI) is a mechanism to allow us to expand MIDI with new features while protecting backward compatibility with MIDI Devices that do not understand these newly defined features.

MIDI-CI separates older MIDI products from newer products with new capabilities and provides a mechanism for two MIDI Devices to understand what new capabilities are supported.

MIDI-CI assumes and requires bidirectional communication. Once a MIDI-CI connection is established between Devices, query and response messages define what capabilities each Device has. MIDI-CI then negotiates or auto-configures to use those features that are common between the Devices.

MIDI-CI provides test mechanisms when enabling new features. If a test fails, then Devices fall back to using MIDI 1.0 for that feature.

MIDI-CI improves MIDI capabilities in several key areas. MIDI-CI allows Devices to use an expanded MIDI protocol with high resolution and multiple per note controllers. It allows for incremental adoption of new MIDI features by providing a fallback to MIDI 1.0 Devices in all cases.

MIDI-CI Includes Queries for 3 major areas of expanded MIDI functionality:

1. Protocol Negotiation
2. Profile Configuration
3. Property Exchange

http://www.midi.org/

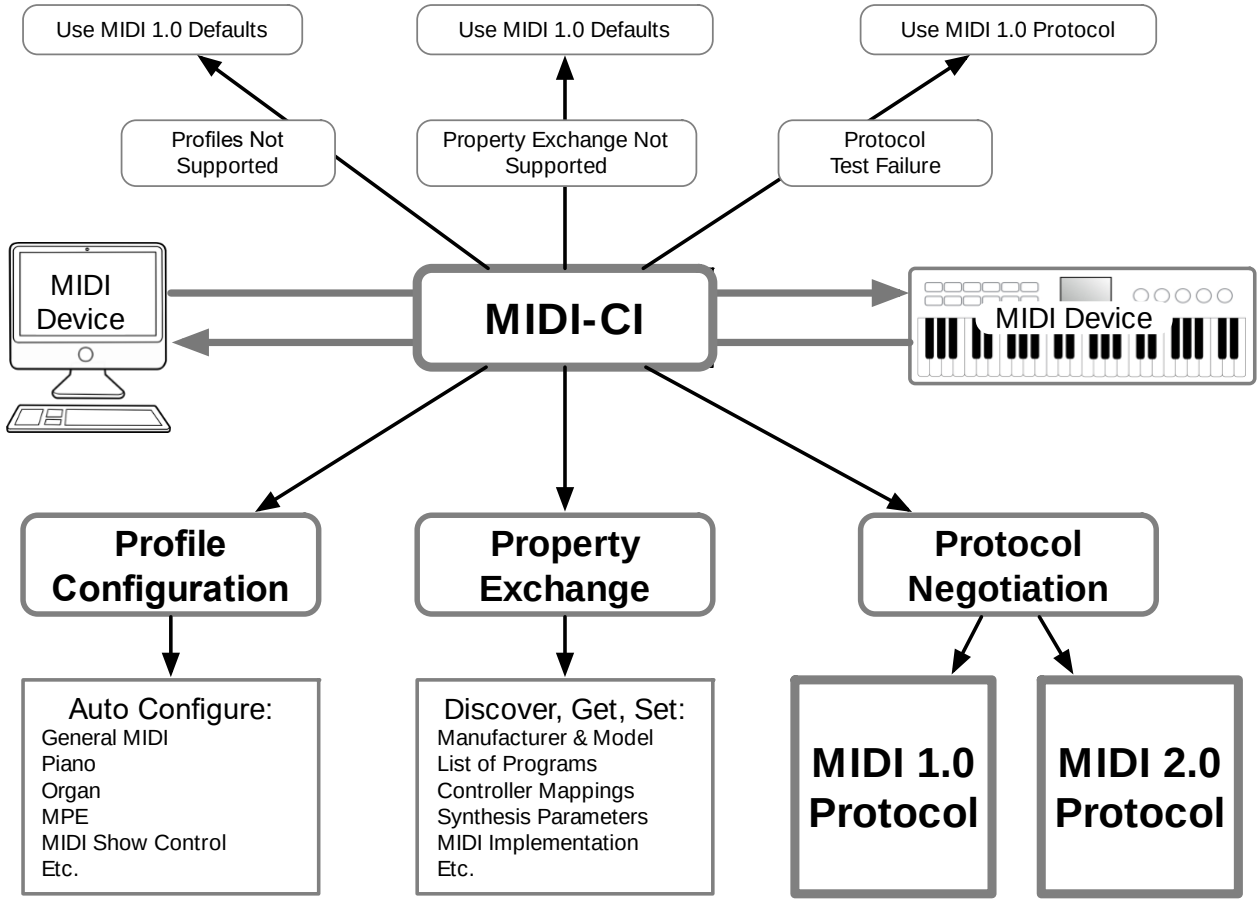
Table of Contents

1. INTRODUCTION

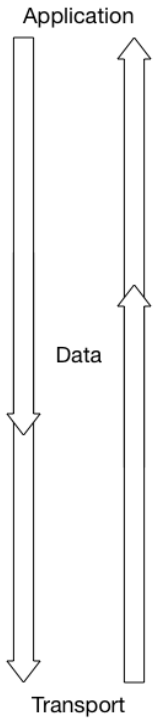
1.1 Background

MIDI-CI defines an architecture that allows Devices with bidirectional communication to agree to use extended MIDI capabilities beyond those defined in MIDI 1.0, while carefully protecting backward compatibility. MIDI-CI features "fall back" mechanisms so that if a Device does not support new features MIDI continues to work as defined by MIDI 1.0. Goals of MIDI-CI design:

1. Fully backward compatible: supports continued MIDI 1.0 functionality for any Devices that do not recognize extended MIDI features enabled by MIDI-CI.
2. Allow easy configuration between MIDI-CI Devices.
3. Sender can know the capabilities of a Receiver.
4. Sender and Receiver can negotiate auto-configuration details.
5. Define method for negotiating choice of Protocol between Devices.
6. Define method for using Profiles.
7. Define method for Discovering, Getting, and Setting a wide range of Device Properties.



1.2 Layer Model for MIDI-CI

	MIDI-CI INQUIRY & NEGOTIATION	MIDI LAYER	DEFINITION	EXAMPLES	MIDI-CI FUNCTION
	PROPERTY EXCHANGE	**DEVICE**	Manufacturer and Model Specific Details	Products sold to the users of MIDI	MIDI-CI allows get and set for a wide range of properties or state of a Device.
		MIDI IMPLEMENTATION	Channels and Messages Supported	Controller Mappings	MIDI-CI allows discovery of MIDI implementation details of a Device.
	PROFILE CONFIGU-RATION	**PROFILES**	A collection of defined Device Parameters and MIDI Messages common across manufacturers	General MIDI, MPE, Hi-Res Piano Profile (to come soon)	MIDI-CI allows general auto-configuration of MIDI Implementation between MIDI Devices that share common Profiles.
	PROTOCOL NEGOTIATION	**PROTOCOL**	Data Language (MIDI 1.0 Messages with 1 Status Bit and 7 Data Bits or extended MIDI 2.0 Messages)	Note-On, Control Change, Program Change	MIDI-CI allows selection of MIDI 1.0 Protocol or "MIDI 2.0 Protocol" with or without timestamps, or added functionality.
	none	**PACKET FORMAT**	Container for Protocol Payload	MIDI 1.0 Stream, USB-MIDI 32 bit Message, BLE-MIDI Packet, Universal MIDI Packet	None (handled by Transport)
	none	**BANDWIDTH**	Data Flow Rate, Throughput, Speed	31.25Kbs on 5pinDIN, 31.25Kbs on USB, 1Mbs on USB	None (handled by Transport)
	DISCOVERY	**TRANSPORT**	Hardware or Software Connection Medium	5PinDIN cable, USB, New 2 Way UART, Ethernet, OS API, VST, CoreMIDI	A MIDI-CI Discovery establishes the pairing of an Input and Output for Bidirectional Communication between Devices, independent of hardware or software connection medium.

1.3 Related Documents:

1. MIDI 1.0 Detailed Specification, Document Version 4.2 September 1995
2. USB Device Class Definition for MIDI Devices, Version 1.0
3. Specification for MIDI over Bluetooth Low Energy, Version 1.0
4. RTP Payload Format for MIDI (IETF RFC 6295)

1.4 Future Pending Documents:

MIDI-CI defines a foundational structure for expanding MIDI. It does not define the expansions themselves. The expansions of MIDI that MIDI-CI enables will be defined in future documents of the MMA and AMEI. These documents may include the following and more:

1. MIDI 2.0 Specification, Version 1.0
2. Universal MIDI Packet and MIDI 2.0 Protocol Specification - Adds new features to existing MIDI messages and defines new MIDI messages.
3. Common Rules for MIDI-CI Profiles - Defines rules for all Profile Specifications
4. Individual Profile Specifications - Define implementation requirements for compliant Devices
5. Common Rules for MIDI-CI Property Exchange - Defines Property Data semantics for Property Exchange
6. Approved Resource Definitions – Define sets of Property Data used by Property Exchange
7. USB Device Class Definition for MIDI Devices, Version 2.0

1.5 Terminology

Chunk – A single System Exclusive message that is one segment of a complete Property Exchange message which spans multiple System Exclusive messages.

Device – A hardware unit or software component.

Endpoint – MIDI Endpoint.

Initiator – One of two MIDI-CI Devices with a bidirectional communication between them. Initiator has the management role of setting and negotiating parameters for interoperability between the two Devices. The primary goal of Initiator is usually (but not strictly required to be) configuring two Devices for subsequent communication from Initiator as MIDI transmitter to Responder as MIDI receiver.

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

MIDI 1.0 Protocol – Version 1.0 of the MIDI Protocol as originally specified in [MMA01]. The native format for the MIDI 1.0 Protocol is a byte stream, but it has been adapted for many different transports. The UMP format for MIDI 1.0 Messages is defined in Section 4 of this specification.

MIDI 2.0 Protocol – Version 2.0 of the MIDI Protocol, as defined in this specification. The native format for MIDI 2.0 messages is UMP as defined in Section 4 of this specification.

MIDI-CI Device – A Device that has the ability to act as a Responder that replies to inquiries received from an Initiator. The ability to act as an Initiator is recommended but optional.

MIDI Endpoint – 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 Event Processor (Sequencer, Arpeggiator) – Records, Edits, and Plays messages or Transforms messages in real time**.** Supports only MIDI 1.0 or Switchable between MIDI 1.0 and MIDI 2.0 Protocol messages on a port by port basis.

MIDI Gateway – A special purpose embedded OS Device (e.g., Workstation, Router) that manages a Node of Connected Devices/Applications/Plugins. Provides connection & routing between all Endpoints on the Node. Acts as MIDI-CI Proxy for Endpoints whenever necessary. Supports only MIDI 1.0 or is switchable between MIDI 1.0 and MIDI 2.0 Protocol messages.

MIDI Node Server (PC) – General purpose OS, with wide range of MIDI service and MIDI API e.g., Mac or Windows PC. Manages a Node of Connected Devices/Applications/Plugins. Provides connection & routing between all Endpoints on the Node. Acts as MIDI-CI Proxy for Endpoints whenever necessary. May act as MIDI-CI Proxy for Single Direction Endpoints (i.e. API as Proxy for Plugin) Supports both MIDI 1.0 and MIDI 2.0 Protocol messages. It is strongly recommended that a Central MIDI Node PC have protocol translation capability on every Input/Output MIDI-CI connection.

MIDI Port – A physical connector associated with a MIDI Endpoint. Some people may consider a MIDI Port to be synonymous with a MIDI Endpoint. A MIDI Port always has a MIDI Endpoint. But a MIDI Endpoint does not always have a (physical) MIDI Port; It may have a Virtual MIDI Port instead. Inside software a MIDI Port is a virtual representation of a MIDI Port. In this case it sometimes called a Virtual MIDI Port. When using MIDI-CI on a system that uses the Universal MIDI Packet format, definitions in this specification which refer to a MIDI Port shall apply to a Group of the Universal MIDI Packet format (See the Universal MIDI Packet and MIDI 2.0 Protocol Specification).

MIDI-CI Proxy – A MIDI-CI Device, such as a MIDI Node Server, MIDI Gateway, or MIDI Translator, that represents another MIDI Device in a MIDI-CI Transaction. While acting as a MIDI-CI Proxy, a Device reports the MUID, manufacturer and other fields with values from another Device that it is representing and not its own.

MIDI Translator – Located between Bidirectional Endpoints**.** Performs Translation between MIDI 1.0 and MIDI 2.0 Protocol whenever necessary. Acts as MIDI-CI Proxy for Endpoints whenever necessary. Passes both MIDI 1.0 and MIDI 2.0 Protocol messages.

MIDI Transport – Carries MIDI data between Bidirectional Endpoints**.** Acts as MIDI-CI Proxy for Endpoints whenever necessary. Passes both MIDI 1.0 and MIDI 2.0 Protocol messages. Does NOT do any protocol translation.

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

Profile – A set of MIDI messages and defined responses to those messages. A Profile may have a defined minimum set of mandatory messages, along with some optional or recommended messages. General MIDI is one example of a profile (although in original form it does not quite meets all requirements of a MIDI-CI Profile). General MIDI defines a requirement for supporting GM messages on all 16 channels of 1 virtual cable (or stream). Future Profiles may define support on only 1 channel within a virtual cable, or support for more than 16 channels using multiple virtual cables or Groups.

Protocol – A defined data message structure that defines the semantics for MIDI control messages. In MIDI 1.0 the Protocol includes the Opcode of the control message being sent, system-wide messages, addressing for some messages in the form of 16 MIDI Channels, and for some types of message a value associated with the specific Opcode.

Property Data – The whole set of properties that make up a reply to a Resource inquiry/request. By default, such properties are presented in JSON key:value pairs but a Resource may define other data formats.

Resource – A defined set of properties that comprise a set of Property Data.

Responder – One of two MIDI-CI Devices with a bidirectional communication between them. The Responder is the Device that receives an Inquiry message from an Initiator Device as part of a MIDI-CI Transaction and acts based on negotiation messages managed by an Initiator Device.

Single Direction Endpoint – A MIDI Device that only sends or receives and is, therefore, NOT capable of MIDI-CI. In some cases, a MIDI Node Server, MIDI Gateway, MIDI Translator, or MIDI Transport may "know" details about a Device from a non-MIDI source (USB Device Descriptors, PlugIn API properties, etc.). The MIDI Node Server, MIDI Gateway, MIDI Translator, or MIDI Transport may act as a MIDI-CI Proxy: it engages in MIDI-CI Transactions in place of the Single Direction Endpoint Device.

Transaction – A set of MIDI-CI messages that include an Inquiry sent by an Initiator Device and a reply to the Inquiry returned by the Responder. The Responder's reply to an Inquiry might be a single message that satisfies the Inquiry, a set of multiple messages that satisfy the Inquiry, or an error message.

USB Endpoint – The source or sink of data sent over USB. This is a physical Device with a buffer. A typical USB-MIDI Interface has 3 USB Endpoints: 1 bidirectional Control Endpoint, and 2 USB-MIDI Endpoints (one in each direction) that each can support up to 16 virtual cables with 16 channels each (256 channels).

USB-MIDI Endpoint – A USB Endpoint used to transfer MIDI Data. In version 1.0 of the USB-MIDI specification, a USB Endpoint supports 16 virtual cables worth of MIDI data, by using a 32 bit MIDI Event packet.

1.6 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:

Words Relating to Specification Conformance:

Word	Reserved For	Relation to Spec 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 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:

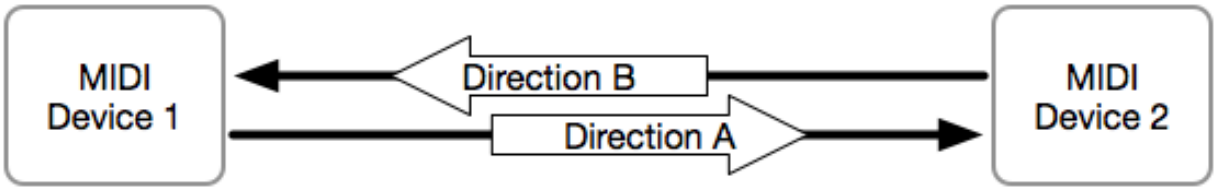
Words Not Relating to Specification Conformance:

Word	Reserved For	Relation to Spec 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. TOPOLOGY

2.1 Bidirectional

MIDI-CI requires bidirectional MIDI communications.



Every MIDI-CI capable Input Port shall be paired with a matching Output Port. Devices may have multiple pairs of MIDI Ports for Input and Output.
Each pair of Ports (1 Input + 1 Output) is used for MIDI-CI Transactions.
See Appendix C for topology limitations related to MIDI Thru and MIDI merge functions.

2.2 Initiator and Responder Relationship

MIDI-CI assumes that MIDI communications tend to be receiver centric; MIDI-CI assumes a system where a MIDI transmitter "learns" something about the receiver and adapts its output as much as possible to support the capabilities of the receiver.
However, through MIDI-CI Negotiation mechanisms, a transmitter can also ask a receiving Device to enable features reported as supported capabilities to adapt the Receiver to the capabilities of the Sender.
In a MIDI-CI bidirectional connection, both Devices are senders and both Devices are receivers for various MIDI-CI messages that constitute a Transaction. Therefore, Initiator and Responder are terms used to clarify the relationship between the two MIDI Devices with a bidirectional connection.
Either of the two Devices may choose to function as the Initiator. The Initiator takes on the management role of setting and negotiation of parameters in the Transaction for interoperability between the two Devices. The primary goal of Initiator is usually (but not strictly required to be) configuring two Devices for subsequent communication from Initiator as MIDI transmitter to Responder as MIDI receiver.

2.3 Bidirectional Negotiation for Bidirectional Settings

Some MIDI-CI Transactions make settings that are common to both directions. In these cases, when the Initiator determines and controls negotiation of particular settings, it does so for equal and identical subsequent interoperability in both directions.

MIDI-CI Protocol Negotiation causes simultaneous changes to the protocol used in both directions. In the diagram in Section 2.1, regardless of whether MIDI Device 1 or MIDI Device 2 is the Initiator, one MIDI-CI Protocol Negotiation establishes a chosen protocol in both Direction A and Direction B.

2.4 Bidirectional Negotiation for Single Direction Settings

Some MIDI-CI Transactions make settings for chosen interoperability in just one direction. In these cases, when the Initiator determines and controls negotiation of particular settings, it does so for subsequent interoperability of messages sent from Initiator to the Responder.

By default, MIDI-CI Transactions for Profile Configuration and Property Exchange are intended for discovering and making settings for interoperability in just one direction, from Initiator to Responder.

When a Device chooses to be Initiator, it takes on the management role of setting parameters in the Transaction for MIDI interoperability from Initiator to Responder. If the Device that is Responder also wants to set parameters in the opposite direction, then the Responder switches roles and becomes Initiator to manage settings for communication in the opposite direction.

> Example: MIDI Device 1 starts as Initiator Device by sending a Profile Inquiry and the MIDI Device 2 acts as Responder and answers with a Reply to Profile Inquiry. Then MIDI Device 1 may send messages to enable and disable Profiles that on MIDI Device 2. If MIDI Device 2 wishes to control Profiles on MIDI Device 1 (that is the opposite direction), then it shall take on the role of Initiator, and initiate a Profile Inquiry.

2.5 Selecting Initiator for Bidirectional Negotiation for Bidirectional Settings

When two Devices share a bidirectional connection to each other, either Device may choose to act in the Initiator role for any Transaction. In some cases, both Devices want to act as the Initiator.

This is not a problem for Bidirectional Negotiation for Single Direction Settings (specifically Profile Negotiation and Property Exchange).

However, when MIDI-CI uses a Bidirectional Negotiation for Bidirectional Settings (specifically Protocol Negotiation), there can be a conflict between the two Devices simultaneously vying to manage the connection.

The following rules allow Devices to assign the Initiator role to one of two connected Devices in a Bidirectional Negotiation for Bidirectional Settings (see Section 2.3). In this version of MIDI-CI, these rules apply only to Protocol Negotiation. These do not apply to Profile Configuration or Property Exchange. Apply the rules in the following order:

1. First Inquiry

The MIDI-CI Device that first sends a MIDI-CI inquiry assumes the role of Initiator. The MIDI-CI Device that receives the initial inquiry assumes the role of Responder.

2. Authority Level Overrides First Inquiry (See Section 3.2.2)

A Device may optionally refuse to act as a Responder only if it has a higher Authority Level as reported in an Authority Level field of a message.

When a 1st MIDI-CI Device takes the role of Initiator, a 2nd MIDI-CI Device may refuse to act as a Responder if the 2nd MIDI-CI Device has a higher Authority Level than the 1st Device's Authority Level. Instead of sending a Responder reply message, the 2nd MIDI-CI Device may send an initial inquiry message to claim Initiator role. When the 1st Device receives that inquiry in reply from the 2nd Device using the same Source MUID as the 1st Device's inquiry's Destination MUID, it shall change its role to Responder.

3. Simultaneous Inquiries = Use Authority Level (See Section 3.2.2)

In some cases, both Devices may try to initiate a MIDI-CI Transaction at the same time. In those cases, the Devices shall use their Authority Level fields to determine which Device continues as Initiator.

When a Device sends an initial inquiry message it takes the role of Initiator. If the Device receives an initial inquiry message at the same time as or following its own initial inquiry message, it shall compare its own Authority Level to the Authority Level of the incoming message. If the incoming message has a higher Authority Level, the Device shall set its role to Responder and send a reply to the inquiry message it received.

If the Device's own Authority Level to the Authority Level of the incoming message are the same, the Device will use the MUID of the two Devices to assign authority according to the following rule.

4. Same Authority Level = Use Highest MUID (See Section 3.2.3)

When two Devices simultaneously try to take the Initiator role and both Devices are of the same Authority Level, the Device that has the highest value for MUID shall take the role of Initiator.

The Device that sent the lower MUID shall set its role to Responder and send a reply to the inquiry message it received.

2.5.1 Authority Level

MIDI-CI provides an Authority Level, a 1 byte field with integer value, to designate Devices that have management authority (the Initiator role) as compared to other Devices. Devices are prioritized in the following order (highest integer value = highest level of authority):

0x70-0x7F Reserved

0x60-0x6F Highest Authority Level

0x50-0x5F

0x40-0x4F

0x30-0x3F

0x20-0x2F

0x10-0x1F Lowest Authority Level

0x00-0x0F Reserved

Descriptions and requirements of those Devices follow. These are not strict requirements, sometimes it is difficult to classify a Device to any defined type. These are just recommendations to help Device manufacturers decide what value to assign to the Device's Authority Level field.

While these are just recommendations, no Device shall claim the highest level of authority if the Device does not provide a wide range of MIDI services and API to manage multiple connected Devices as are typically found in a general-purpose PC.

AUTHORITY LEVEL	RECOMMENDED FOR THESE TYPICAL DEVICES
0x70-0x7F	Reserved
0x60-0x6F Highest Authority Level	MIDI Node Server (PC)
0x50-0x5F	MIDI Gateway
0x40-0x4F	MIDI Translator
0x30-0x3F	MIDI Endpoint
0x20-2F	MIDI Event Processor (Sequencer, Arpeggiator)

0x10-1F Lowest Authority Level	MIDI Transport
0x00-0x0F	Reserved

2.5.2 User Selected Initiator

Some Devices may allow a user to trigger a MIDI-CI inquiry from that Device. In most cases, other Devices should allow the Device of user's choice to take authority.

> One typical example implementation might be a "MIDI-CI AutoConfiguration" button on the front panel of a Device. This is useful for a controller keyboard connected to a DAW and several plugins. As the user changes plugins, the "MIDI-CI AutoConfiguration" performs dynamic configuration changes on the controller keyboard to match the state or functions of the currently selected plugin.

The primary application for User Selected Initiator is to perform Profile Configuration and Property Exchange.

The Device manufacturer should take care in implementation of this function to avoid triggering unnecessary system changes. In particular, this function may perform Protocol Negotiation once upon startup, but it should not trigger Protocol Negotiation again until a power cycle or there has been a topology change that necessitates a new Protocol Negotiation.

2.6 MIDI-CI Proxy Device

MIDI Node Servers, MIDI Gateways, and MIDI Translators manage connections and pass MIDI streams to different ports or via different protocols. Example Devices include a MIDI API in a general-purpose computer, Plugin API, MIDI-DIN to USB-MIDI converters, MIDI processors (hardware or in software), DAWs with multiple Plugins, or networked MIDI bridges like RTP-MIDI. There are multiple options for if and how MIDI Node Servers, MIDI Gateways, and MIDI Translators may engage in the MIDI-CI connection.

> 1. If a MIDI Node Server, MIDI Gateway, or MIDI Translator merely passes the MIDI stream through, it should act as a transparent bridge and pass on the MIDI-CI messages without modification. Then MIDI-CI enabled Devices on both ends of the MIDI Node Server, MIDI Gateway, or MIDI Translator will be able to establish a direct connection to each other.
>
> 2. Sometimes a MIDI Node Server, MIDI Gateway, or MIDI Translator may act as a **MIDI-CI Proxy** for other Devices. It reports the manufacturer and other fields with values from the other Device and not its own. A MIDI Node Server, MIDI Gateway, or MIDI Translator shall not misrepresent the Device; fields should only be populated with known values (e.g. from USB descriptors or Plugin API properties). A MIDI Node Server, MIDI Gateway, or MIDI Translator shall not guess values or use arbitrary default values. Some example applications include:
>
> - If a MIDI Node Server or MIDI Gateway is managing routing between various Devices, it may need to inform connected Devices of routing changes. The MIDI Node Server or MIDI Gateway may act as a MIDI-CI Proxy to inform Devices of Profile changes triggered by connection or topology changes.
> - If a Device that is NOT capable of MIDI-CI is connected to a MIDI Node Server, MIDI Gateway, or MIDI Translator, and if the MIDI Node Server, MIDI Gateway, or MIDI Translator has some knowledge about that Device, the MIDI Node Server, MIDI Gateway, or MIDI Translator should act as a MIDI-CI Proxy; it engages in MIDI-CI Negotiation in place of that Device.

3. In some cases, it makes sense for a MIDI Node Server, MIDI Gateway, or MIDI Translator to engage in the MIDI-CI Negotiation as itself. For example, if it modifies the MIDI data stream in a way that is incompatible with MIDI-CI such as converting to a non-MIDI protocol, or merging/duplicating MIDI streams.

In all cases, it is up to the manufacturer to decide the most sensible approach. MIDI Node Servers, MIDI Gateways, and MIDI Translators which can be configured by the user may offer different options to be selected by the user.

3. Message Addressing and MUID

MIDI-CI messages are exchanged between Devices using addresses or routing determined by a combination of:

1. An established Bidirectional MIDI connection
2. System Exclusive Device ID Fields
3. MIDI-CI Devices' MUIDs

3.1 System Exclusive Device ID Field

Values in the Universal System Exclusive Device ID are used for Channel addressing. Values 0x00-0x0F map to MIDI Channels 1-16. Messages with a Value of 0x7F are addressed to the whole MIDI port/cable, not channelized. Whether this field refers to the Source or Destination depends on the definition of each message type. See Section 5.3 and individual messages for more details.

3.2 MIDI-CI Device's MUID

MUID is 28 bit random number generated by a MIDI-CI Device used to uniquely identify messages to or from that Device. All MIDI-CI messages include the MUID of the source Device and the MUID of the destination Device.
The value of the MUID shall be in the range 0x00000000 to 0x0FFFFFFF.
The values 0x0FFFFF00 to 0x0FFFFFFE are reserved.
The value 0x0FFFFFFF is used as a Broadcast MUID (see Section 3.2.2).

3.2.1 Generating a MUID

Every time a MIDI-CI Device is powered up, it shall create a new, randomly generated MUID. The MIDI-CI Device shall use this same MUID for every Transaction until the Device is shut down and restarted or until it receives an Invalidate MUID message (see Section 5.7) for that MUID. When a Device is restarted it shall generate a new MUID.

A MIDI-CI Device shall not use the same MUID every time it restarts.

The chances of a collision of MUID, where more than one Device on a MIDI connection selects the same MUID, is greatly reduced if all MIDI-CI Devices use good random number generators for their MUIDs (see Appendix B). In the rare case that a collision does occur, MIDI-CI provides rules and mechanisms for resolving the collision (see Section 3.2.3).

3.2.2 Broadcast MUID

Certain MIDI-CI messages are defined to use a Broadcast MUID as a replacement for a specific MUID. This Broadcast MUID is used because the sender of a MIDI-CI message does not know the MUID of a potential receiver, or because the intended destination of the message is several Devices.

Broadcast MUID = 0x0FFFFFFF

Messages sent to the Broadcast MUID shall not be larger than 512 bytes or shall have a defined chunking mechanism so the buffers of any connected receivers will not overflow.

The Broadcast MUID shall be used only for the following MIDI-CI Messages:

- Discovery
- Invalidate MUID
- Profile Enabled Report

- Profile Disabled Report
- Profile Specific Data (If defined as allowed by the active Profile)

Broadcast MUID shall not be used as the destination for any MIDI-CI message unless the definition for that message specifically defines the use of the Broadcast MUID.

3.2.3 Potential Collisions of MUID

Even if every MIDI-CI Device uses a good random number generator for its MUID, there is the rare possibility that two Devices might select the same MUID. Section 5.7.1 defines mechanisms to resolve collisions.

4. Establishing a MIDI-CI Connection

To begin using MIDI-CI, a pair of MIDI-CI Devices find each other using a Discovery Transaction.

4.1 The First MIDI-CI Transaction: Discovery

A MIDI-CI Device shall find another MIDI-CI Device by acting as an Initiator and sending a Discovery message (see Section 5.5) with its own MUID as the source. Any MIDI-CI Device that receives the Discovery Message shall act as a Responder by sending a Reply to Discovery message (see Section 5.6) with its own MUID as the source and the Initator's MUID as the destination.

4.2 Subsequent Transactions

Following a successful Discovery Transaction between two Devices, either Device may take the role of Initiator for any subsequent Transactions between the two Devices by sending a MIDI-CI Inquiry to the destination MUID of a targeted Responder. For any subsequent Transaction, the Responder is a Device that receives a MIDI-CI Inquiry that was sent to the Responders MUID, acts based on the content of the inquiry sent by an Initiator Device, and responds to the Initiator with a reply message.

5. MIDI-CI COMMON RULES AND GUIDELINES

This section outlines concepts and rules common to all categories of MIDI-CI messages and Transactions.

5.1 Categories of MIDI-CI Messages

MIDI-CI defines Universal System Exclusive messages, Transactions, and mechanisms for the inquiry and negotiation of Device capabilities in four Categories. Each Category contains multiple messages that work together to deliver a targeted scope of MIDI-CI functionality.

The Category is declared by the value of the highest nibble in the Universal System Exclusive Sub-ID#2 byte.

Category	Sub-ID#2 Range	Description
0	0x00-0x0F	Reserved – No Messages Defined Yet
1	0x10-0x1F	**Protocol Negotiation Messages**
2	0x20-0x2F	**Profile Configuration Messages**
3	0x30-0x3F	**Property Exchange Messages**
4	0x40-0x4F	Reserved – No Messages Defined Yet
5	0x50-0x5F	Reserved – No Messages Defined Yet
6	0x60-0x6F	Reserved – No Messages Defined Yet
7	0x70-0x7F	**Management Messages**

5.2 MIDI-CI Transactions - Order of Processing

Devices do not need to support all Categories of MIDI-CI implementation.

The first time Devices establish a MIDI-CI connection, the Devices shall proceed through Transactions in the following order for any Categories of MIDI-CI that they support.

1. Discovery Transaction
2. Protocol Negotiation
3. Profile Configuration (Profile Inquiry Transaction)
4. Property Exchange (Inquiry: Property Exchange Capabilities Transaction)

If either the Initiator or Responder do not support a Category of MIDI-CI Inquiry and Negotiation, then the Devices shall proceed with the next Category of MIDI-CI.

After this first set of Transactions take place, a Device may freely use any Inquiry or Negotiation message as required and independently from other Categories or messages of Inquiry and Negotiation.

5.3 MIDI-CI Transaction Messages

MIDI-CI Transactions are accomplished using Universal System Exclusive messages.

Message layout tables in this specification show the 0xF0 SysEx Start and 0xF7 SysEx End bytes, which are required when using the MIDI 1.0 data format. In some AMEI/MMA protocols, including those carried inside a Universal MIDI Packet, the F0 and F7 are omitted.

All messages conform to a common format with header and data as follows.

5.3.1 Standard Format for MIDI-CI Messages

All MIDI-CI messages use 0x0D as value for Universal System Exclusive Sub-ID#1.

The Universal System Exclusive Sub-ID#2 determines the Category of message and the function of each message.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Device ID: Source or Destination (depending on type of message) 7F = to/from MIDI Port 00-0F = to/from MIDI Channels 1-16 10-7E = Reserved
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
1 byte	Universal System Exclusive Sub-ID#2: Category and Type of MIDI-CI Message 0x00-0F Reserved 0x10-1F Protocol Negotiation Messages 0x20-2F Profile Configuration Messages 0x30-3F Property Exchange Messages 0x40-6F Reserved 0x70-7F Management Messages
1 byte	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first*)
4 bytes	Destination MUID (LSB first*)
nb bytes	Data. Includes any necessary fields to suit the needs of each type of MIDI-CI message.
F7	End Universal System Exclusive

*Note: Multibyte fields are generally LSB First unless defined by MIDI 1.0 as a byte sequence.

Device ID: Source or Destination (depending on type of message)

Values in this Universal System Exclusive Device ID are used for Channel addressing. Values 0x00-0xF map to MIDI Channels 1-16. Messages with a Value of 0x7F are addressed to the whole MIDI Port, not channelized.

Some MIDI-CI messages, such as Protocol Negotiation messages, are only valid when sent to a whole MIDI Port 0x7F. See individual messages as defined in this document for whether each message can be used on a per channel basis.

In a MIDI-CI inquiry message sent by the Initiator, this is the destination of the inquiry. Default value is 0x7F = MIDI Port based inquiry. But value can be 0x00-0x0F to address a specific MIDI Channel.

In a MIDI-CI reply message sent by the Responder, this is the source of the reply. Default value is 0x7F = MIDI Port based reply. But the value can be 0x00-0x0F to reply about a specific MIDI Channel.

Values 0x00-0x0F are for 16 MIDI channels. This allows inquiries and negotiation on a specific channel. This is useful for using Profile Configuration and Property Exchange on a per channel basis. Values 0x00-0x0F shall not be used for Protocol Negotiation messages.

1 byte MIDI-CI Message Version/Format

In this 1.1 version of the MIDI-CI specification, the version number is 0x01.

4 bytes Source MUID

The MUID of the Device sending this message.

4 bytes Destination MUID

The MUID of the Device intended to receive this message.

Data

If the message contains any payload it is in this field. Some messages define multiple fields in the Data field.

5.4 MIDI-CI Messages Format and Protocols

MIDI-CI enables switching between various protocol types. This MIDI-CI specification defines MIDI-CI messages using a MIDI Universal System Exclusive message format for use with any protocol that supports System Exclusive.

If MIDI-CI is used to negotiate to an AMEI / MMA standard protocol that does not support System Exclusive, that protocol shall define equivalent MIDI-CI messages using native messages of that protocol.

5.5 Discovery Message

An Initiator shall establish connections to other MIDI-CI Devices by sending a Discovery message. The Discovery message also declares the MUID of the Initiator. A Discovery message may be sent in response to various events on the Initiator Device. Some examples of when a Discovery message might be sent include:

- A MIDI-CI capable Device should send a Discovery message after its power on and boot up procedure is complete.
- A MIDI-CI capable Device should send a Discovery message when the user selects the MIDI-CI Start button or similar autoconfiguration function on a MIDI Device.
- The Application Programming Interface (API) and/or Driver for MIDI services on a host computer should notify all applications on the host when a new MIDI Device is connected and discovered. The

API or applications should send a Discovery message to auto-configure interoperability with the new Device.

- The Device has had it previous MUID invalidated and wants to re-establish MIDI-CI connections using a new MUID.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	Device ID: 7F = to MIDI Port
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
70	Universal System Exclusive Sub-ID#2: Discovery
1 byte	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
7F 7F 7F 7F	Destination MUID (LSB first) (to Broadcast MUID)
3 bytes	Device Manufacturer (System Exclusive ID Number)
2 bytes	Device Family (LSB first)
2 bytes	Device Family Model Number (LSB first)
4 bytes	Software Revision Level (Format is Device specific)
1 byte	Capability Inquiry Category Supported (bitmap)
4 bytes	Receivable Maximum SysEx Message Size (LSB first)
F7	End Universal System Exclusive

Four fields for Device Identification
The four fields described below identify the Device using the same data as defined by the "Device Inquiry" Universal System Exclusive message (See MIDI 1.0 Detailed Specification). The data is formatted as follows:

3 bytes Device Manufacturer
This is the System Exclusive ID of the Device manufacturer. For System Exclusive ID values that are only 1 byte in length, the System Exclusive ID value is in the first byte and the remaining 2 bytes are filled with zeroes: ID 00 00

2 bytes Device Family
This identifies the related group of models to which the Device belongs. The manufacturer is free to determine the grouping of models and the format of the data in this field.

2 bytes Device Family Model Number
This identifies a specific model from the Device Manufacturer. The manufacturer is free to determine the assignment of values and the format of the data in this field.

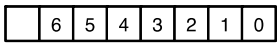
4 bytes Software Revision Level

This is the version number of a Device model number. This is typically version of software or firmware but may also be version of hardware. If a model undergoes any version update or other design change that changes its midi implementation or capabilities as may be discovered by MIDI-CI (including Property Exchange), then this Software Revision Level shall be changed. The manufacturer is free to determine the format of the data in this field.

Capability Inquiry Category Supported

This field is a bitmap which reports the Categories of MIDI-CI messages the Device supports. A value which is set high in any bit indicates support for some inquiry messages of that Category. Support for all messages of that Category are not required to declare support.

Capability Inquiry Category Supported Bitmap



The bitmap contains bits corresponding to the high nibble of the Sub-ID#2 of various MIDI-CI messages.

Bit	Category	Supported Sub-ID#2 Range	Description
0	0	0x00-0x0F	Reserved – No Messages Defined Yet
1	1	0x10-0x1F	**Protocol Negotiation Supported**
2	2	0x20-0x2F	**Profile Configuration Supported**
3	3	0x30-0x3F	**Property Exchange Supported**
4	4	0x40-0x4F	Reserved – No Messages Defined Yet
5	5	0x50-0x5F	Reserved – No Messages Defined Yet
6	6	0x60-0x6F	Reserved – No Messages Defined Yet
7	none	none	Most significant bit of MIDI Data Byte. This is always set to zero.

See Appendix D for a list of all MIDI-CI messages in all Categories.

Note: Management Messages (Sub-ID#2 value=0x70-0x7F) shall be supported by all Devices that implement MIDI-CI. This MIDI-CI Discovery message and the associated reply (following) that contain this field are both of that Management Message Category.

Receivable Maximum SysEx Message Size

1. All MIDI-CI Devices shall support System Exclusive message lengths of at least 128 bytes. The allowed values in the Receivable Maximum SysEx Message Size are 128 or greater.

2. MIDI-CI Devices that have the ability to Initiate Transactions for Profile Configuration or Property Exchange categories shall support message lengths of at least 512 bytes. The allowed values in the Receivable Maximum SysEx Message Size are 512 or greater.

5.5.1 Timeout for Discovery

After sending a Discovery Message, an Initiator shall wait at least 3 seconds for all Reply to Discovery Messages to be returned before timing out.

5.6 Reply to Discovery Message

When a MIDI-CI Device receives a Discovery message it shall become a Responder and send this Reply to Discovery message. This message declares the MUID of the Responder.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	Device ID: 7F = from MIDI Port
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
71	Universal System Exclusive Sub-ID#2: Reply to Discovery
1 byte	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
3 bytes	Device Manufacturer (System Exclusive ID Number)
2 bytes	Device Family (LSB first)
2 bytes	Device Family Model Number (LSB first)
4 bytes	Software Revision Level (Format is Device specific)
1 byte	Capability Inquiry Category Supported
4 bytes	Receivable Maximum SysEx Message Size (LSB first)*
F7	End Universal System Exclusive

*See the Receivable Maximum SysEx Message Size in Section 5.5

Note: This Reply to Discovery message is mandatory for all MIDI-CI Devices. Devices which do not support any MIDI-CI Categories may optionally send a this reply to inform the Initiator of the existence of a device which does not support any MIDI-CI functions.

5.7 Invalidate MUID Message

An Invalidate MUID message has 2 applications:

- A Device should send an Invalidate MUID message when it is shutting down or if for any other reason the Device will not continue to use its MUID.
- An Invalidate MUID messages is used in mechanisms for resolving collisions of MUID when 2 or more Devices declare the same MUID (see Section 5.7.1).

Every MIDI-CI Device shall process received Invalidate MUID messages.

If a Device receives an Invalidate MUID message with the Target MUID set to the same value as its own MUID, it shall terminate any active Transactions and generate a new MUID.

If a Device receives an Invalidate MUID message with the Target MUID set to the same value as any other Devices it has previously discovered, it shall terminate any active Transactions with that MUID and should discard all cached information of Devices with the invalidated MUID.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	Device ID: 7F = to MIDI Port
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
7E	Universal System Exclusive Sub-ID#2: Invalidate MUID
1 byte	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
7F 7F 7F 7F	Destination MUID (LSB first) (to Broadcast MUID)
4 bytes	Target MUID (the MUID to Invalidate) (LSB first)
F7	End Universal System Exclusive

When sending or receiving an Invalidate MUID message, a Device is not required to disable any enabled Profiles or revert to previous Protocols. An Invalidate MUID message immediately ends all current, pending, or outstanding Transactions that are using the Target MUID, including Property Exchange inquiries, replies, and Subscriptions.

When a Device receives an Invalidate MUID message, it does not send any reply or confirmation message.

5.7.1 Resolving Collisions of MUID - Responder

If a Responder receives a MIDI-CI Discovery message with the Source MUID set to the same value as its own MUID, then the Responder shall select one of two options to resolve the collision:

Option A, only applicable if the Responder has not yet used its MUID in any prior Transactions:

1. The Responder shall change its own MUID to a new value.
2. The Responder shall send a Reply to Discovery message with the new MUID value.

Option B:

1. The Responder shall reply with an Invalidate MUID message with the Target MUID set to the duplicated MUID.
2. The Responder shall change its own MUID to a new value.
3. The Initiator shall change its own MUID to a new value.
4. Any Device or all Devices may Initiate a new Discovery Transaction

5.7.2 Resolving Collisions of MUID - Initiator

If an Initiator sends a MIDI-CI Discovery message and receives multiple replies and where two or more of the Responders have the same MUID as each other, then:

1. The Initiator shall send an Invalidate MUID message with the Target MUID set to the duplicated MUID.
2. All Devices with the duplicated MUID shall change their MUID to a new value.
3. Any Device or all Devices may Initiate a new Discovery Transaction

5.8 NAK MIDI-CI Message

The MIDI-CI NAK message is used to respond to any message that a Device does not understand. Examples of application for this NAK message include:

- Reply to a MIDI-CI message the Device does not support
- Reply to a MIDI-CI message with MIDI-CI message Version/Format the Device does not support*
- Reply to a malformed MIDI-CI message
- Reply to a Profile Enable or Disable message for a Profile the Responder does not support or does not support on the requested channel.

Receiver response to a NAK message is undefined.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Device ID: Source 7F = from MIDI Port 00-0F = from MIDI Channels 1-16 10-7E = Reserved
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
7F	Universal System Exclusive Sub-ID#2: MIDI-CI NAK
1 byte	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
F7	End Universal System Exclusive

6. PROTOCOL NEGOTIATION

This mechanism selects a new protocol for communication between MIDI-CI Devices. The Protocol selected for use between MIDI-CI Devices is common to both directions of a bidirectional connection. If Protocol used is changed in one direction, the Protocol in the opposite direction is changed at the same time.

6.1 Protocol Types Supported

MIDI-CI enables switching to protocols supported by two Devices connected to each other. The choice of protocols available in this revision of MIDI-CI are:

0x01 MIDI 1.0
0x02 MIDI 2.0 Protocol

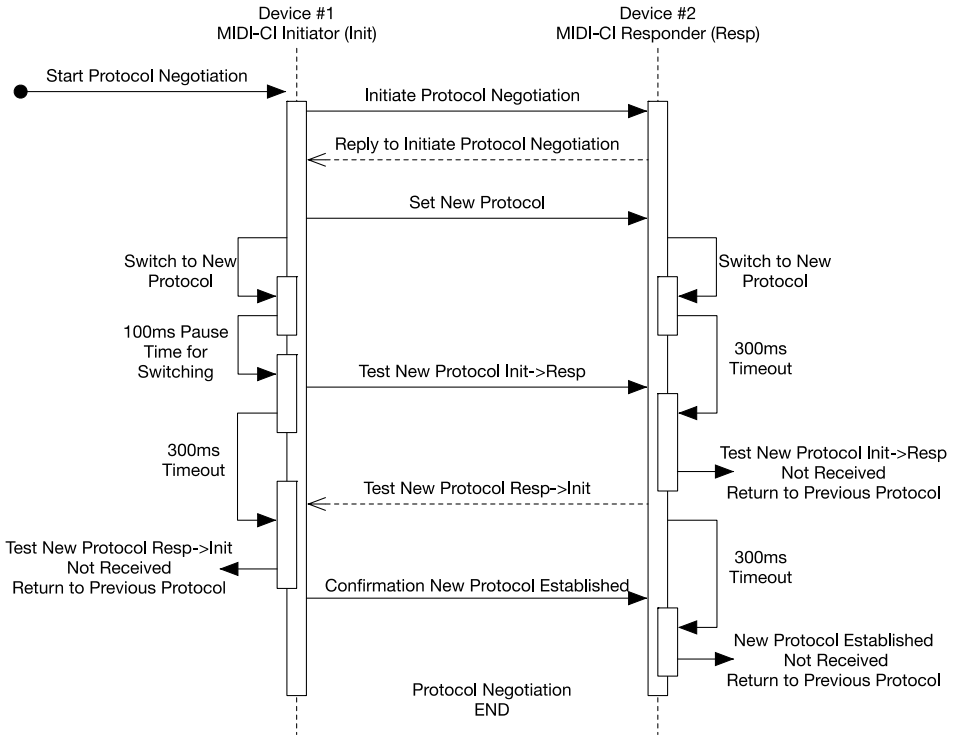
Values 0x00, 0x03-0x7F are Reserved

6.2 Universal MIDI Packet Required

MIDI-CI Protocol Negotiation is dependent on the use of the Universal MIDI Packet format. If MIDI Devices are not connected by a transport that supports the Universal MIDI Packet format, the Protocol Negotiation should not be initiated. If it is initiated, the Protocol Negotiation has a very high probability of failure.

6.3 Protocol Inquiry and Negotiation Mechanism

Initiator shall begin Protocol Negotiation with Initiate Protocol Negotiation. When Responder is ready to switch Protocols, it shall reply with Responder Reply Protocol Capabilities message. At this point Initiator may decide a Timeout before escaping negotiation or restarting the negotiation. Other Timeout and escapes are defined and shown in the following diagram.



In case of failure, both Devices shall always return to “Previous Protocol”. In many cases Previous Protocol will be MIDI 1.0. But in some cases, Devices might be using some other Protocol (Example MIDI 2.0 Protocol) and then start a negotiation to yet another Protocol. If that negotiation fails, the return to Previous Protocol is not a return to MIDI 1.0 (Example MIDI 2.0 Protocol).

6.4 Initiate Protocol Negotiation Message

This initial message is both an Inquiry and a Report of Initiator capabilities.

<table>
<tr><th>Value</th><th>Parameter</th></tr>
<tr><td>F0</td><td>System Exclusive Start</td></tr>
<tr><td>7E</td><td>Universal System Exclusive</td></tr>
<tr><td>7F</td><td>To/From whole MIDI Port</td></tr>
<tr><td>0D</td><td>Universal System Exclusive Sub-ID#1: MIDI-CI</td></tr>
<tr><td>10</td><td>Universal System Exclusive Sub-ID#2: Initiate Protocol Negotiation</td></tr>
<tr><td>01</td><td>MIDI-CI Message Version/Format</td></tr>
<tr><td>4 bytes</td><td>Source MUID (LSB first)</td></tr>
<tr><td>4 bytes</td><td>Destination MUID (LSB first)</td></tr>
<tr><td>1 byte</td><td>Authority Level</td></tr>
<tr><td>1 byte</td><td>Number of Supported Protocols (np)</td></tr>
<tr><td>5 bytes</td><td>Preferred Protocol Type:
<table>
<tr><td>Protocol Byte 1</td><td>0x01=MIDI 1.0</td><td>0x02=MIDI 2.0</td></tr>
<tr><td>Protocol Byte 2</td><td>Version</td><td>Version</td></tr>
<tr><td>Protocol Byte 3</td><td>Extensions</td><td>Extensions</td></tr>
<tr><td>Protocol Byte 4</td><td>Reserved *1
Set to 0x00</td><td>Reserved *1
Set to 0x00</td></tr>
<tr><td>Protocol Byte 5</td><td>Reserved *1
Set to 0x00</td><td>Reserved *1
Set to 0x00</td></tr>
</table>
*1: Reserved field value is 0x00 (null). Other values may be defined in future specifications.</td></tr>
<tr><td rowspan="3">(np-1)x5 bytes</td><td>Optional: Another Supported Protocol in 5 bytes.</td></tr>
<tr><td>…</td></tr>
<tr><td>Optional: Last Supported Protocol</td></tr>
<tr><td>F7</td><td>End Universal System Exclusive</td></tr>
</table>

1 byte Authority Level

See Section 2.5.1.

Supported Protocols:

Each Protocol supported by a Device is listed with a set of 5 bytes (Protocol Byte 1 - Protocol Byte 5). The Protocol that is preferred by the Device should be the first in the list of Supported Protocols.

All Devices shall support MIDI 1.0 as one of the Protocol choices. If a Device only supports one Protocol, it shall be MIDI 1.0.

The preferred choice of Protocol for the Device shall be the first one listed. Other Protocols supported shall be listed in order of preference.

Using MIDI 1.0 Protocol

Protocol Byte 1, Protocol Type

The type number for MIDI 1.0 is 0x01.

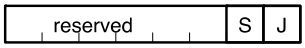
Protocol Byte 2, Version

The version number for MIDI 1.0 is 0x00.

Protocol Byte 3, Extensions

If the 2 Devices agreeing to a MIDI-CI Protocol Negotiation are connected by a transport that supports the Universal MIDI Packet format, then there are defined extensions available for using MIDI 1.0. The Extensions field is a bitmap of extension flags or optional features. At time of writing this version of MIDI-CI, there are 2 extensions defined. Further extensions may be defined by the Association of Musical Electronics Industry and the MIDI Manufacturers Association in future revisions of MIDI-CI or in the MIDI 2.0 Protocol specification.

MIDI 1.0 Protocol Extensions Bitmap Field



S = Size of Packet extension flag. When MIDI 1.0 Devices use the Universal MIDI Packet format they shall be capable of handling messages of up to 64 bits in size. When S=0, message packets exchanged shall not exceed 64 bits in size. When S=1, message packets of 96 bits and 128 bits in size may also be exchanged. This larger size is necessary to support SysEx 8 and Mixed Data Set messages.

J = Jitter Reduction Timestamps extension flag. When J=1 then Jitter Reduction Timestamps are supported and shall be used preceding every MIDI 1.0 Protocol message.

When a Device reports S=0 and J = 1, then the Device shall be able to handle messages up to 64 bits in size with 32 additional bits for JR Timestamps for a total combined size of 96 bits.

When a Device reports S=1 and J = 1, then the Device shall be able to handle messages of 128 bits in size with 32 additional bits for JR Timestamps for a total combined size of 160 bits.

Using MIDI 2.0 Protocol

Protocol Byte 1, Protocol Type

The type number for MIDI 2.0 Protocol is 0x02.

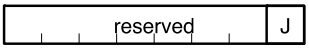
Protocol Byte 2, Version

The version number for Version 1.0 of MIDI 2.0 Protocol is 0x00.

Protocol Byte 3, Extensions

The Extensions field is a bitmap of extension flags or optional features. At time of writing this version of MIDI-CI, the only extension defined at the time of this specification release is Jitter Reduction Timestamps. Further extensions may be defined by the Association of Musical Electronics Industry and the MIDI Manufacturers Association in future revisions of MIDI-CI or in the MIDI 2.0 Protocol specification.

MIDI 2.0 Protocol Extensions Bitmap Field



J = Jitter Reduction Timestamps extension flag. When J=1 then Jitter Reduction Timestamps are supported and shall used preceding every MIDI 2.0 Protocol message.

Note: When Devices use MIDI 2.0 Protocol in the Universal MIDI Packet format, they shall be capable of handling messages of up to 128 bits in size. When a Device reports J = 1, then the Device shall be able to handle messages of 128 bits in size with 32 additional bits for JR Timestamps for a total combined size of 160 bits.

6.5 Reply to Initiate Protocol Negotiation Message

<table>
<tr><th>Value</th><th>Parameter</th></tr>
<tr><td>F0</td><td>System Exclusive Start</td></tr>
<tr><td>7E</td><td>Universal System Exclusive</td></tr>
<tr><td>7F</td><td>To/From whole MIDI Port</td></tr>
<tr><td>0D</td><td>Universal System Exclusive Sub-ID#1: MIDI-CI</td></tr>
<tr><td>11</td><td>Universal System Exclusive Sub-ID#2: Reply to Initiate Protocol Negotiation</td></tr>
<tr><td>01</td><td>MIDI-CI Message Version/Format</td></tr>
<tr><td>4 bytes</td><td>Source MUID (LSB first)</td></tr>
<tr><td>4 bytes</td><td>Destination MUID (LSB first)</td></tr>
<tr><td>1 byte</td><td>Authority Level</td></tr>
<tr><td>1 byte</td><td>Number of Supported Protocols (np)</td></tr>
<tr><td>5 bytes</td><td>Preferred Protocol Type:

<table>
<tr><td>Protocol Byte 1</td><td>0x01=MIDI 1.0</td><td>0x02=MIDI 2.0</td></tr>
<tr><td>Protocol Byte 2</td><td>Version</td><td>Version</td></tr>
<tr><td>Protocol Byte 3</td><td>Extensions</td><td>Extensions</td></tr>
<tr><td>Protocol Byte 4</td><td>Reserved *1
Set to 0x00</td><td>Reserved *1
Set to 0x00</td></tr>
<tr><td>Protocol Byte 5</td><td>Reserved *1
Set to 0x00</td><td>Reserved *1
Set to 0x00</td></tr>
</table>
*1: Reserved field value is 0x00 (null). Other values may be defined in future specifications.</td></tr>
<tr><td rowspan="3">(np-1)x5 bytes</td><td>Optional: Another Supported Protocol in 5 bytes.</td></tr>
<tr><td>...</td></tr>
<tr><td>Optional: Last Supported Protocol</td></tr>
<tr><td>F7</td><td>End Universal System Exclusive</td></tr>
</table>

Supported Protocols:

The Responder should reply with a list of all Protocols that it can support, in order of preference of the Responder. However, the Responder may optionally adapt its list of supported Protocols to leave out protocols that are not supported by the Initiator (as reported in the Initiate Protocol Negotiation message).

6.6 Set New Protocol Message

The Initiator selects new Protocol based on matching its own capabilities against the capabilities of the Responder. Initiator sends the newly selected Protocol to the Responder.

<table>
<tr><th>Value</th><th>Parameter</th></tr>
<tr><td>F0</td><td>System Exclusive Start</td></tr>
<tr><td>7E</td><td>Universal System Exclusive</td></tr>
<tr><td>7F</td><td>To/From whole MIDI Port</td></tr>
<tr><td>0D</td><td>Universal System Exclusive Sub-ID#1: MIDI-CI</td></tr>
<tr><td>12</td><td>Universal System Exclusive Sub-ID#2: Set New Selected Protocol</td></tr>
<tr><td>01</td><td>MIDI-CI Message Version/Format</td></tr>
<tr><td>4 bytes</td><td>Source MUID (LSB first)</td></tr>
<tr><td>4 bytes</td><td>Destination MUID (LSB first)</td></tr>
<tr><td>1 byte</td><td>Authority Level</td></tr>
<tr><td>5 bytes</td><td>New Protocol Type:
<table>
<tr><td>Protocol Byte 1</td><td>0x01=MIDI 1.0</td><td>0x02=MIDI 2.0</td></tr>
<tr><td>Protocol Byte 2</td><td>Version</td><td>Version</td></tr>
<tr><td>Protocol Byte 3</td><td>Extensions</td><td>Extensions</td></tr>
<tr><td>Protocol Byte 4</td><td>Reserved *1
Set to 0x00</td><td>Reserved *1
Set to 0x00</td></tr>
<tr><td>Protocol Byte 5</td><td>Reserved *1
Set to 0x00</td><td>Reserved *1
Set to 0x00</td></tr>
</table>
*1: Reserved field value is 0x00 (null). Other values may be defined in future specifications.</td></tr>
<tr><td>F7</td><td>End Universal System Exclusive</td></tr>
</table>

After the Initiator sends this Set New Protocol message, it shall switch its own Protocol while also waiting 100ms to allow the Responder to switch Protocol. The Initiator shall then send the next message, Test New Protocol Initiator to Responder. (100ms is a guideline)

After the Responder receives this Set New Protocol message, it shall switch its own Protocol. It also starts a 300ms timeout with expectation of receiving the Test New Protocol Initiator to Responder message from the Initiator. (300ms is a guideline)

6.7 Test New Protocol Initiator to Responder Message

The Initiator sends this confirmation test message in the new Protocol. The Responder uses this to confirm that the Protocol has been successfully established between Initiator MIDI Out and Responder MIDI In.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	To/From whole MIDI Port
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
13	Universal System Exclusive Sub-ID#2: Test New Protocol Initiator to Responder
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
1 byte	Authority Level
48 bytes	Test Data: string of 48 numbers in ascending order: 0x00, 0x01, 0x02 ... 0x2E, 0x2F.
F7	End Universal System Exclusive

After the Initiator sends this test message, it shall start a 300ms timeout counter with expectation of receiving a Test New Protocol Responder to Initiator message from the Responder. (300ms is a guideline)

After the Responder successfully receives this test message, it shall reply with the next message, the Test New Protocol Responder to Initiator.

If the Responder does not successfully receive this test message before its 300ms timeout counter expires, it shall reset its Protocol to the previous value. (300ms is a guideline)

6.8 Test New Protocol Responder to Initiator Message

The Responder sends this confirmation test message in the new Protocol. The Initiator uses this to confirm that the Protocol has been successfully established between Initiator MIDI Out and Responder MIDI In (via previous test) and between Responder MIDI Out and Initiator MIDI In.

Value	Parameter

F0	System Exclusive Start
7E	Universal System Exclusive
7F	To/From whole MIDI Port
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
14	Universal System Exclusive Sub-ID#2: Test New Protocol Responder to Initiator
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
1 byte	Authority Level
48 bytes	Test Data: string of 48 numbers in ascending order: 0x00, 0x01, 0x02 ... 0x2E, 0x2F.
F7	End Universal System Exclusive

After the Initiator successfully receives this test message, it shall reply with the next message, the Confirmation New Protocol Established.

If the Initiator does not successfully receive this test message before its 300ms timeout counter expires, it shall reset its Protocol to the previous value. Then the Initiator can decide whether to restart Protocol Negotiation. (300ms is a guideline)

6.9 Confirmation New Protocol Established Message

The Initiator sends this confirmation test message in the new Protocol. The Responder uses to confirm that the Protocol has been successfully established.

Value	**Parameter**
F0	System Exclusive Start
7E	Universal System Exclusive
7F	To/From whole MIDI Port
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
15	Universal System Exclusive Sub-ID#2: Confirmation New Protocol Established
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
1 byte	Authority Level

F7	End Universal System Exclusive

Ongoing MIDI messages may now be exchanged bidirectionally between the two Devices using the newly established Protocol.

6.10 Subsequent Protocol Negotiation

If two Devices have used Protocol Negotiation to successfully negotiate to another Protocol and want to negotiate again to another protocol (such as a negotiated return to MIDI 1.0) then the entire Protocol Negotiation process shall be restarted. There is one difference in the process:

> It is assumed that Protocol Negotiation generally starts between two MIDI Devices using MIDI 1.0 messages. However, after a Protocol Negotiation, the two Devices may be using another Protocol. Therefore, Protocol Negotiation shall start using the negotiation messages in the current Protocol (not always MIDI 1.0)

Also see Section 5.4

7. PROFILE CONFIGURATION

Profiles define specific implementations of a set of MIDI messages chosen to suit a particular instrument, Device type, or to accomplish a particular task. Two Devices that conform to the same Profile will have generally have greater interoperability between them than Devices using MIDI without Profiles. Profiles increase interoperability and ease of use while reducing the amount of manual configuration of Devices by users.

7.1 Profile Configuration Mechanism

Profiles are controlled by the following Common Profile Configuration messages:

- Profile Inquiry
- Reply to Profile Inquiry
- Set Profile On
- Set Profile Off
- Profile Enabled Report
- Profile Disabled Report

More information about Profiles is defined in other specifications of MMA and AMEI.

7.2 Profile Inquiry Message

An Initiator may send this to request a list of Profiles that a connected Responder Device supports.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Destination 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
20	Universal System Exclusive Sub-ID#2: Profile Inquiry
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
F7	End Universal System Exclusive

7.3 Reply to Profile Inquiry Message

When a Responder receives the Profile Inquiry message it shall reply with this message to report a list of Profiles the Responder supports.

There are 2 lists of Supported Profiles in the message:

1. Profiles that are Supported and Currently Enabled
2. Profiles that are Supported but Currently Disabled

The Initiator may use this information to auto-configure the connection between the Devices for increased interoperability.

<table>
<tr><th>Value</th><th>Parameter</th></tr>
<tr><td>F0</td><td>System Exclusive Start</td></tr>
<tr><td>7E</td><td>Universal System Exclusive</td></tr>
<tr><td>1 byte</td><td>Source
7F = to/from whole MIDI Port
00-0F = to/from MIDI Channels 1-16</td></tr>
<tr><td>0D</td><td>Universal System Exclusive Sub-ID#1: MIDI-CI</td></tr>
<tr><td>21</td><td>Universal System Exclusive Sub-ID#2: Reply to Profile Inquiry</td></tr>
<tr><td>01</td><td>MIDI-CI Message Version/Format</td></tr>
<tr><td>4 bytes</td><td>Source MUID (LSB first)</td></tr>
<tr><td>4 bytes</td><td>Destination MUID (LSB first)</td></tr>
<tr><td>2 bytes</td><td>Number of Currently-Enabled Profiles (cep) (LSB first)</td></tr>
<tr><td>5 bytes</td><td>Profile ID of First Currently-Enabled Profile:
<table>
<tr><td>Profile ID
Byte 1</td><td>0x7E Standard
Defined Profile</td><td>Manufacturer SysEx ID 1 Profile</td></tr>
<tr><td>Profile ID
Byte 2</td><td>Profile Bank</td><td>Manufacturer SysEx ID 2 Profile</td></tr>
<tr><td>Profile ID
Byte 3</td><td>Profile Number</td><td>Manufacturer SysEx ID 3 Profile</td></tr>
<tr><td>Profile ID
Byte 4</td><td>Profile Version</td><td>Manufacturer Specific Info</td></tr>
<tr><td>Profile ID
Byte 5</td><td>Profile Level</td><td>Manufacturer Specific Info</td></tr>
</table></td></tr>
<tr><td>(cep - 1)
x 5 bytes</td><td>Optional: Profile ID of Other Currently-Enabled Profiles in sets of 5 bytes.
...
Optional: Profile ID of Last Currently-Enabled Profile</td></tr>
<tr><td>2 bytes</td><td>Number of Currently-Disabled Profiles Supported (cdp) (LSB first)</td></tr>
<tr><td>5 bytes</td><td>Profile ID of First Currently-Disabled Profile Supported</td></tr>
<tr><td>(cdp - 1)</td><td>Optional: Profile ID of Other Currently-Disabled Profiles Supported in sets of 5 bytes.</td></tr>
</table>

x 5 bytes	... Optional: Profile ID of Last Currently-Disabled Profile Supported
F7	End Universal System Exclusive

Profile ID

Each Profile has a 5 byte identifier. Standard Defined Profiles are those adopted by AMEI and MMA. Each one uses the ID defined by each Profile Specification and by other AMEI/MMA Profile related specifications. The value of the Profile ID Byte 1 is 0x7E (Universal)

Manufacturers may use MIDI-CI to control Profiles of their own proprietary design by using their own System Exclusive ID. For System Exclusive ID values that are only 1 byte in length, the System Exclusive ID value is in the first byte and the remaining 2 bytes are filled with zeroes.

Each Manufacturer SysEx ID can freely use the 2 bytes of Manufacturer Specific Info. These 2 bytes allow up to 16384 different Manufacturer Profile Numbers.

Profile ID Byte 1	0x7E Standard Defined Profile	Manufacturer SysEx ID 1 Profile
Profile ID Byte 2	Profile Bank	Manufacturer SysEx ID 2 Profile
Profile ID Byte 3	Profile Number	Manufacturer SysEx ID 3 Profile
Profile ID Byte 4	Profile Version	Manufacturer Specific Info
Profile ID Byte 5	Profile Level	Manufacturer Specific Info

Currently Enabled

These are Profiles that the Device supports and that are currently active at the time of inquiry. Some Devices might have a Profile that is already active (Enabled) before receiving a Set Profile On message. For example, a MIDI acoustic piano might be fixed to conform to a Piano Profile Specification; Piano Profile is always Enabled.

Currently Disabled

These are Profiles that a Device can support but that are not currently active. When a Profile on a Device is not active (Disabled), the Device does not currently conform to the requirements of the Profile specification. But the Device can be switched to conform to the requirements of the Profile specification using the Set Profile On message.

Note: If a Device does not support any Profiles, then the Device may send this Reply to Profile Inquiry message with the Number of Currently-Enabled Profiles set to 0x00 and with the Number of Currently-Disabled Profiles Supported set to 0x00.

7.4 Set Profile On Message

An Initiator may send this to enable a Profile on a Responder.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Destination 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
22	Universal System Exclusive Sub-ID#2: Set Profile On
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
5 bytes	Profile ID of Profile to be Set to On (to be Enabled)
F7	End Universal System Exclusive

7.5 Set Profile Off Message

An Initiator may send this to disable a Profile on a Responder.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Destination 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
23	Universal System Exclusive Sub-ID#2: Set Profile Off
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
5 bytes	Profile ID of Profile to be Set to Off (to be Disabled)
F7	End Universal System Exclusive

7.6 Profile Enabled Report Message

A Device shall send this message if it has enabled a Profile.
This is an acknowledgement upon receipt of a Set Profile On message.
This is an informative message if any other event enables a Profile.

A Device shall send this message if it is unable to comply with a Set Profile Off message and the Profile remains enabled.

The Profile Enabled Report message shall always have the Destination MUID field set to the Broadcast MUID.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Source 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
24	Universal System Exclusive Sub-ID#2: Profile Enabled
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
7F 7F 7F 7F	Destination MUID (LSB first) (to Broadcast MUID)
5 bytes	Profile ID of Profile that is now Enabled
F7	End Universal System Exclusive

7.7 Profile Disabled Report Message

A Device shall send this message if it has disabled a Profile.
This is an acknowledgement upon receipt of a Set Profile Off message.
This is an informative message if any other event disables a Profile.

A Device shall send this message if it is unable to comply with a Set Profile On message and the Profile remains disabled.

The Profile Disabled Report message shall always have the Destination MUID field set to the Broadcast MUID.

Value	Parameter
F0	System Exclusive Start

7E	Universal System Exclusive
1 byte	Source 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
25	Universal System Exclusive Sub-ID#2: Profile Disabled
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
7F 7F 7F 7F	Destination MUID (LSB first) (to Broadcast MUID)
5 bytes	Profile ID of Profile that is now Disabled
F7	End Universal System Exclusive

7.8 Profile Specific Data Message

Some Profile specifications might need to define some System Exclusive messages to support unique features or to communicate data relating to that Profile ID. This message allows a Profile to send data that is specific to the Profile ID without the need for a separately assigned Universal SysEx Sub ID. Profile specifications may make use of this message and may freely define the contents of the Profile Specific Data field.

Value	**Parameter**
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Source 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
2F	Universal System Exclusive Sub-ID#2: Profile Specific Data
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)*
5 bytes	Profile ID
4 bytes	Length of Following Profile Specific Data (LSB first)
nn bytes	Profile Specific Data
F7	End Universal System Exclusive

* Profile Specific Data messages shall not be sent to the Broadcast MUID unless the Profile Specification defines the use of the Broadcast MUID. Messages sent to the Broadcast MUID shall not be larger than 512 bytes or shall have a defined chunking mechanism so the buffers of any connected receivers will not overflow.

For future definition of a Profile Specific Data message, the designer should consider the application and the capabilities of targeted receiver types. Some simple Responders might not be able to receive any message larger than 128 bytes. See “Receivable Maximum SysEx Message Size” in Section 5.5.

8. PROPERTY EXCHANGE

Property Exchange is used to Inquire, Get, and Set many properties including but not limited to Device configuration settings, a list of controllers and resolution, a list of patches with names and other metadata, manufacturer, model number, and version.
Detailed information about the Header Data and Property Data used in Property Exchange messages is defined in the Common Rules for Property Exchange specification. The MIDI-CI specification contains only the base messages used for Property Exchange. Most of the definition for implementing Property Exchange, including all defined properties and semantics, exists outside of the MIDI-CI specification.

8.1 Property Inquiry and Negotiation Mechanism

Properties are exchanged by the following Common Property Exchange messages:

- Inquiry: Property Exchange Capabilities
- Reply to Property Exchange Capabilities
- Inquiry: Has Property Data
- Reply to Has Property Data
- Inquiry: Get Property Data
- Reply to Get Property Data
- Inquiry: Set Property Data
- Reply to Set Property Data
- Subscription
- Reply to Subscription
- Notify

8.2 Property Data May Be Sent in Multiple Chunks

A Device may choose to send a Property Exchange message as a single SysEx message or as a set of multiple SysEx messages or "Chunks".
When a complete Property Exchange message, with all defined SysEx fields and the payload Property Data, exceeds the size of the "Receivable Maximum SysEx Message Size" of the other Device (discovered in the initial Discovery Transaction between the Devices) the sender shall break the message into multiple Chunks. A Device may also choose to send a message in multiple Chunks for its own design requirements.
If the Device chooses to send Property Data in multiple Chunks, it shall specify the "Number of Chunks in Message" and shall label each Chunk with a sequential "Number of This Chunk". The Number of This Chunk shall always start counting from a value of 0x0001.
If the sender Device does not know the total number of Chunks in advance, the Device shall set the Number of Chunks in Message to 0x0000. Then when sending Chunks, the sender shall set the Number of This Chunk for each Chunk in a sequential count as usual.
When Number of Chunks in Message is unknown, the final Chunk shall declare a new value for Number of Chunks in Message to match the sequential count value of Number of This Chunk.

If a sender runs out of Property Data or otherwise needs to terminate a message before sending the expected number of Chunks to match Number of Chunks in Message, then the final Chunk sent shall indicate the end of data in one of two ways.

1. If the sender knows that all the Property Data sent is complete or usable, then the sender shall change the Number of Chunks in Message to match the Number of This Chunk.
2. If the sender does not know if all the Property Data sent is complete or usable, then the Number of This Chunk for the final Chunk shall be set to 0x0000.

If the sender runs out of Property Data before sending a final Chunk with data, then the sender shall send one more Chunk with no Property Data to complete the data set as defined above.

8.2.1 No Chunking of Header Data

Any message that contains Header Data only and does not contain any Property Data shall not use the Chunking mechanism. For any message that does not contain any Property Data, Number of Chunks in Message shall be set to 1 and Number of This Chunk shall be set to 1.

Chunking Example: Data Sets that Require 6 Chunks:

Total Expected Number of Chunks	For First Chunk to (Final-1 Chunk)		For Final Chunk	
	Number of Chunks in Message	Number of This Chunk	Number of Chunks in Message	Number of This Chunk
Known Number: 6 Property Data is Successful	6	1-5	6	6
Known Number: 6 But Property Data is Bad	6	1-5	6	0
Known Number: 6 But Unexpected End Before Chunk 6, Property Data Remains Good/Valid	6	1-4	5	5
Known Number: 6 But Unexpected End Before Chunk 6, Property Data is Unknown or Bad	6	1-4	5	0
Unknown Number Property Data is Successful	0	1-5	6	6
Unknown Number Property Data is Unknown or Bad	0	1-5	6	0

8.3 Multiple Simultaneous Inquiries and Request ID

A Request ID allows the Device to support multiple messages or PE Transactions being sent and received at one time. This is useful to prevent a larger PE message which is split over many chunks

from blocking smaller requests. Each Device may support one or more Request IDs, with the default being one Request ID. Every Chunk of a message shall contain the same Request ID. The reply to an inquiry message shall contain the same Request ID as was sent in the associated inquiry message.
As the Number of Simultaneous Property Exchange Requests Supported is indicative of the processing power of the Device, the same pool of Request IDs should be used across all PE Inquiry types.
Request ID values are unique only to the connection between a specific Initiator and specific Responder, determined by the MUID of those two Devices. The same Request ID value may be active on a separate MIDI connection between a different pair of MUIDs without incurring a collision.

8.4 Inquiry: Property Exchange Capabilities

An Initiator shall send this to exchange basic information with the Responder before sending and receiving subsequent Property Exchange messages.

This inquiry does not need to be performed for every Property Exchange Transaction. Devices may cache information discovered by this message so that this inquiry might be performed only once after the Discovery Transaction and before starting any other Property Exchange inquiries.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Destination 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
30	Universal System Exclusive Sub-ID#2: Inquiry: Property Data Exchange Capabilities
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
1 byte	Number of Simultaneous Property Exchange Requests Supported
F7	End Universal System Exclusive

8.5 Reply to Property Exchange Capabilities

When a Responder receives the Inquiry: Property Exchange Capabilities message it shall reply with this message to report basic information for sending and receiving PE messages.

Value	Parameter

F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Source 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
31	Universal System Exclusive Sub-ID#2: Reply to Property Data Exchange Capabilities
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
1 byte	Number of Simultaneous Property Exchange Requests Supported
F7	End Universal System Exclusive

8.6 Inquiry: Has Property Data

A Reserved message. Shall not be used until defined by AMEI/MMA.

Value	**Parameter**
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Destination 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
32	Universal System Exclusive Sub-ID#2: Inquiry: Has Property Data
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
1 byte	Request ID
2 bytes	Length of Following Header Data (in this Chunk) (LSB first)
nn bytes	Header Data
2 bytes	Number of Chunks in Message (LSB first) 0x0000 = Number of Chunks in Message is Unknown

2 bytes	Number of This Chunk (count starts from 0x0001) (LSB first) 0x0000 = Final Chunk when Number of Chunks in Message is Unknown
2 bytes	Length of Following Property Data (in this Chunk) (LSB First)
nn bytes	Property Data
F7	End Universal System Exclusive

8.7 Reply to Has Property Data

A Reserved message. Shall not be used until defined by AMEI/MMA.

Value	**Parameter**
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Source 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
33	Universal System Exclusive Sub-ID#2: Reply to Has Property Data
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
1 byte	Request ID
2 bytes	Length of Following Header Data (in this Chunk) (LSB first)
nn bytes	Header Data
2 bytes	Number of Chunks in Message (LSB first) 0x0000 = Number of Chunks in Message is Unknown
2 bytes	Number of This Chunk (count starts from 0x0001) (LSB first) 0x0000 = Final Chunk when Number of Chunks in Message is Unknown
2 bytes	Length of Following Property Data (in this Chunk) (LSB first)
nn bytes	Property Data
F7	End Universal System Exclusive

8.8 Inquiry: Get Property Data

An Initiator shall send this to discover Property Data in a receiving Responder.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Destination 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
34	Universal System Exclusive Sub-ID#2: Inquiry: Get Property Data
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
1 byte	Request ID
2 bytes	Length of Following Header Data (in this Chunk) (LSB first)
nn bytes	Header Data
2 bytes	Number of Chunks in Message (LSB first) 0x0000 = Number of Chunks in Message is Unknown
2 bytes	Number of This Chunk (count starts from 0x0001) (LSB first) 0x0000 = Final Chunk when Number of Chunks in Message is Unknown
2 bytes	Length of Following Property Data (in this Chunk) (LSB first)
nn bytes	Property Data*
F7	End Universal System Exclusive

*This message does not include any Property Data as defined in this version of MIDI-CI. Set Length of Following Property Data to 0x0000.

8.9 Reply to Get Property Data

A Responder shall send this reply after receiving an Inquiry: Get Property Data message.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive

1 byte	Source 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
35	Universal System Exclusive Sub-ID#2: Reply to Get Property Data
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
1 byte	Request ID
2 bytes	Length of Following Header Data (in this Chunk) (LSB first)
nn bytes	Header Data
2 bytes	Number of Chunks in Message (LSB first) 0x0000 = Number of Chunks in Message is Unknown
2 bytes	Number of This Chunk (count starts from 0x0001) (LSB first) 0x0000 = Final Chunk when Number of Chunks in Message is Unknown
2 bytes	Length of Following Property Data (in this Chunk) (LSB first)
nn bytes	Property Data
F7	End Universal System Exclusive

8.10 Inquiry: Set Property Data

An Initiator shall send this to set Property Data in a receiving Responder.

Value	**Parameter**
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Destination 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
36	Universal System Exclusive Sub-ID#2: Inquiry: Set Property Data
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)

1 byte	Request ID
2 bytes	Length of Following Header Data (in this Chunk) (LSB first)
nn bytes	Header Data
2 bytes	Number of Chunks in Message (LSB first) 0x0000 = Number of Chunks in Message is Unknown
2 bytes	Number of This Chunk (count starts from 0x0001) (LSB first) 0x0000 = Final Chunk when Number of Chunks in Message is Unknown
2 bytes	Length of Following Property Data (in this Chunk) (LSB first)
nn bytes	Property Data*
F7	End Universal System Exclusive

8.11 Reply to Set Property Data

A Responder shall send this reply after receiving an Inquiry: Set Property Data message.

Value	**Parameter**
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Source 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
37	Universal System Exclusive Sub-ID#2: Reply to Set Property Data
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
1 byte	Request ID
2 bytes	Length of Following Header Data (in this Chunk) (LSB first)
nn bytes	Header Data
2 bytes	Number of Chunks in Message (LSB first) 0x0000 = Number of Chunks in Message is Unknown
2 bytes	Number of This Chunk (count starts from 0x0001) (LSB first) 0x0000 = Final Chunk when Number of Chunks in Message is Unknown

2 bytes	Length of Following Property Data (in this Chunk) (LSB first)
nn bytes	Property Data
F7	End Universal System Exclusive

*This message does not include any Property Data as defined in this version of MIDI-CI. Set Length of Following Property Data to 0x0000.

8.12 Subscription

An Initiator may establish a Subscription to Property Data in a Responder using a Subscription message. Subsequently, the Responder may then send updates for that Property Data via this Subscription messages or may use this message to end the Subscription (depending on Header Data as defined in the Common Rules for Property Exchange).
Note: An Initiator shall not send updates to the Property Data by this message but shall send updates to the Property Data using an Inquiry: Set Property Data message instead.

Value	**Parameter**
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Destination 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
38	Universal System Exclusive Sub-ID#2: Subscription
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
1 byte	Request ID
2 bytes	Length of Following Header Data (in this Chunk) (LSB first)
nn bytes	Header Data
2 bytes	Number of Chunks in Message (LSB first) 0x0000 = Number of Chunks in Message is Unknown
2 bytes	Number of This Chunk (count starts from 0x0001) (LSB first) 0x0000 = Final Chunk when Number of Chunks in Message is Unknown
2 bytes	Length of Following Property Data (in this Chunk) (LSB first)
nn bytes	Property Data

F7	End Universal System Exclusive

8.13 Reply to Subscription

A Device shall send this reply after receiving a Subscription message.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Source 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
39	Universal System Exclusive Sub-ID#2: Reply to Subscription
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
1 byte	Request ID
2 bytes	Length of Following Header Data (in this Chunk) (LSB first)
nn bytes	Header Data
2 bytes	Number of Chunks in Message (LSB first) 0x0000 = Number of Chunks in Message is Unknown
2 bytes	Number of This Chunk (count starts from 0x0001) (LSB first) 0x0000 = Final Chunk when Number of Chunks in Message is Unknown
2 bytes	Length of Following Property Data (in this Chunk) (LSB first)
nn bytes	Property Data
F7	End Universal System Exclusive

8.14 Notify Message

This is an informative message which may be sent by either Initiator or Responder, to report some types of error messages or other information.

Errors are most commonly reported in Reply messages. See the Common Rules for Property Exchange specification for details.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Source 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
3F	Universal System Exclusive Sub-ID#2: Notify
01	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
1 byte	Request ID
2 bytes	Length of Following Header Data (in this Chunk) (LSB first)
nn bytes	Header Data
2 bytes	Number of Chunks in Message (LSB first) 0x0000 = Number of Chunks in Message is Unknown
2 bytes	Number of This Chunk (count starts from 0x0001) (LSB first) 0x0000 = Final Chunk when Number of Chunks in Message is Unknown
2 bytes	Length of Following Property Data (in this Chunk) (LSB first)
nn bytes	Property Data
F7	End Universal System Exclusive

Appendix A: Minimum Requirements

To support MIDI-CI at the very minimum, every MIDI-CI Device shall do all of the following.

1) MIDI-CI Devices shall meet all of these general requirements:
   - Support System Exclusive message lengths of at least 128 bytes
   - Send and Respond to MIDI-CI Discovery messages
   - Respond to Invalidate MUID Messages

2) MIDI-CI Devices shall implement multiple features, as defined in the MIDI-CI specification, in order to properly and fully meet the general requirements listed in 1) above.
   For example: To properly respond to MIDI-CI Discovery message, a MIDI-CI Device must generate its own random MUID, detect any conflict between the MUID of another Device with its own MUID and take steps to resolve the collision (not the only requirements). To Respond to Invalidate MUID Message, a MIDI-CI Device must be able to generate a new, different MUID (not the only requirement).

Even devices that do not implement any Category of MIDI-CI negotiations (Protocol Negotiation, Profile Configuration, or Property Exchange) are encouraged to use MIDI-CI. The central feature of the minimum requirements is the ability to Send and Respond to a MIDI-CI Discovery messages. A MIDI-CI Discovery Transaction informs the MIDI system that a device exists on a connection and provides fundamental, identifying data which is very useful for system configuration.

Appendix B: Avoiding Collisions of MUID

MIDI-CI relies on every Device on a MIDI connection having its own, unique MUID. If a collision of MUID occurs, where more than one Device on a MIDI connection selects the same MUID, MIDI-CI provides rules and mechanisms for resolving the collision (see Section 3.2.3). But it is better if collision is avoided in the first place. Following are some suggestions for avoiding collisions of MUID. None of the suggestions in this appendix are requirements for conformance.

Device Design

The chances of a collision of MUID is greatly reduced if all MIDI-CI Devices use good random number generators for their MUIDs.

There are multiple methods to generate a good random number. Some possible designs may include some of the following mechanisms:

- In any Device with a real time clock or high resolution timer, use current time as input seed to the Random Number Generator. Do not use such a time when powering up, but of the (first) use of an MUID, or another irregular event, to avoid MUID collisions when similar Devices are powered up at the same time.
- Use the low bits of a high frequency timer (megahertz range), e.g. CPU clock cycles.
- Devices with a unique serial number might include it in the RNG seed
- Events and incoming numbers from outside can be used for a RNG seed, too. E.g. data in incoming non-MIDI data (e.g. UDP/TCP source port numbers, hash or CRC on a series of data coming in on different interfaces), time of arrival of incoming data.
- USB and/or network topology can be used to seed the RNG.
- Desktop PC Operating Systems generate good Random Numbers. Software should use system supplied numbers.
- Many high powered MCUs have built in random number generators.
- Use onboard circuitry to generate noise for use as an input seed
- Note that simulation of Random Number Generators that use floating pins or clocks/oscillators with logic gates will not likely generate a random number.

Manufacturer Suggestions to Users

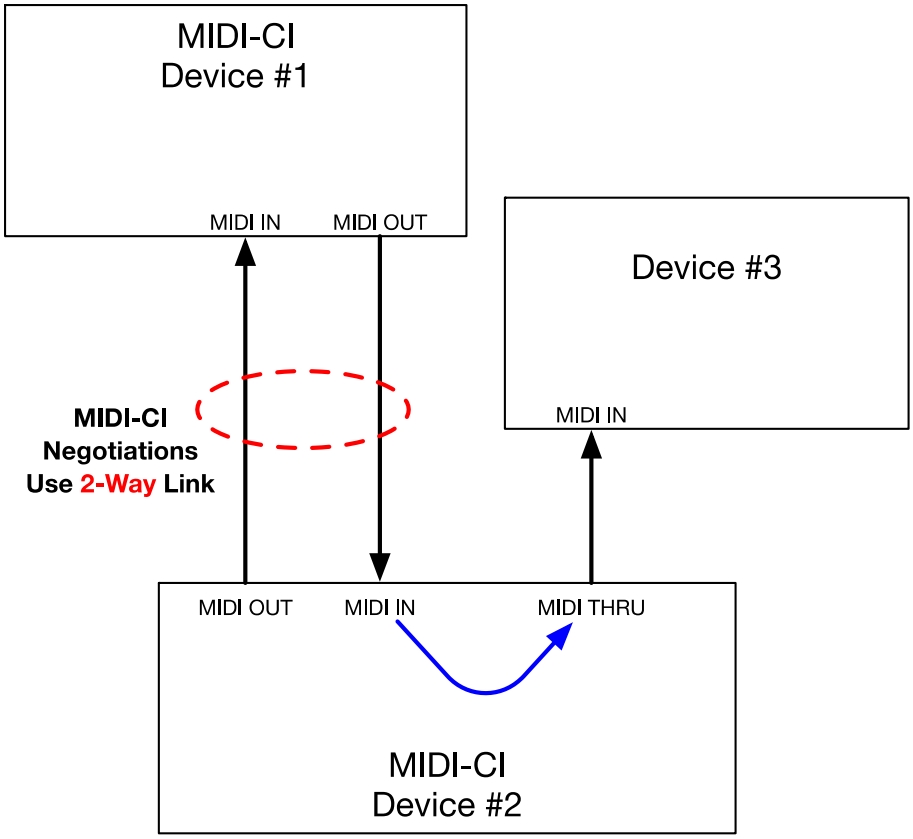
Device manufacturers can help users to avoid the chances of a MUID collision with system configuration advice including the following:

- Connect one Device to one Device. Do not use MIDI Thru. Do not use MIDI merger (especially for low power Devices).
- Connect directly by USB to a Host to get unique addressing for one Device on one Port.
- If you need to merge MIDI streams, use an intelligent Central Hub or Connection Manager that acts as a Device MUID proxy, translating MUID in all messages to/from the colliding Devices.

Appendix C: MIDI Chaining Limitation

It is strongly recommended not to use MIDI Thru ports or a MIDI merger when using MIDI-CI.
Behavior is undefined.

MIDI-CI Negotiations will work reliably on a bidirectional link between two Devices that both support MIDI-CI. MIDI-CI does not support all topologies that MIDI 1.0 allows. MIDI Thru ports add a complication that might cause **significant errors**. A MIDI merger Device may also often introduce significant errors.



Device #3 does not have the 2-way connection necessary for MIDI-CI. It may not be able to support changes that are negotiated between Device #1 and Device #2. Behavior is undefined.

Some MIDI-CI Devices may provide an intelligent MIDI Thru function or an intelligent MIDI merge function that performs any conversions necessary to support such topologies and help mitigate potential incompatibility issues for the user. The details of such a design are not defined in this version of MIDI-CI.

Appendix D: List of All MIDI-CI Messages

List of all MIDI-CI Messages sorted by Universal System Exclusive Sub-ID#2.
Universal System Exclusive Sub-ID#2 declares the Category and Type of MIDI-CI Message:

- 0x00-0F Reserved
- 0x10-1F Protocol Negotiation Messages
- 0x20-2F Profile Configuration Messages
- 0x30-3F Property Exchange Messages
- 0x40-6F Reserved
- 0x70-7F Management Messages

Sub-ID#2	Message Type	Message Source
Category 0: Reserved Space		
0x00-0x0F	Reserved	
Category 1: Protocol Negotiation Messages		
0x10	Initiate Protocol Negotiation Message	Initiator
0x11	Reply to Initiate Protocol Negotiation Message	Responder
0x12	Set New Selected Protocol	Initiator
0x13	Test New Protocol Initiator to Responder	Initiator
0x14	Test New Protocol Responder to Initiator	Responder
0x15	Confirmation Protocol Established	Initiator
0x16-1F	Reserved	
Category 2: Profile Configuration Messages		
0x20	Profile Inquiry	Initiator
0x21	Reply to Profile Inquiry	Responder
0x22	Set Profile On	Initiator
0x23	Set Profile Off	Initiator
0x24	Profile Enabled Report	Initiator or Responder
0x25	Profile Disabled Report	Initiator or Responder
0x26-2E	Reserved	
0x2F	Profile Specific Data	Initiator or Responder
Category 3: Property Exchange Messages		
0x30	Inquiry: Property Exchange Capabilities	Initiator
0x31	Reply to Property Exchange Capabilities	Responder
0x32	Inquiry: Has Property Data(Reserved)	
0x33	Reply to Has Property Data(Reserved)	
0x34	Inquiry: Get Property Data	Initiator
0x35	Reply to Get Property Data	Responder
0x36	Inquiry: Set Property Data	Initiator
0x37	Reply to Set Property Data	Responder
0x38	Subscription	Initiator or Responder
0x39	Reply to Subscription	Initiator or Responder
0x3A-3E	Reserved	
0x3F	Notify	Initiator or Responder

Categories 4-6: Reserved Space		
0x40-6F	Reserved	
Category 7: Management Messages		
0x70	Discovery	Initiator
0x71	Reply to Discovery	Responder
0x72	Invalidate MUID	Initiator or Responder
0x73-7E	Reserved	
0x7F	NAK	Initiator or Responder