MIDI Capability Inquiry (MIDI-CI)

Bidirectional Negotiations for MIDI Devices

MIDI Association Document: M2-101-UM

Document Version 1.2
Draft Date May 11, 2023

Published June 15, 2023

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

PREFACE

MIDI Capability Inquiry (MIDI-CI)

MIDI Capability Inquiry (MIDI-CI) is a set of bidirectional mechanisms which allow devices to negotiate with each other for autoconfiguration. MIDI-CI also allows the expansion of MIDI with new features while protecting backward compatibility with MIDI Devices that do not understand these newly defined features.

AMEI Association of Musical Electronics Industry

http://www.amei.or.jp

MIDI® Association

https://www.midi.org

Version History

Table 1 Version History

Publication Date	Version	Changes
April 26, 2018	1.0	Initial release
February 20, 2020	1.1	Withdrew Session concept including Open and Close session messages. Added Discovery mechanism with MUID. Many other notable changes and many editorial improvements.
June 15, 2023	1.2	Added Process Inquiry. Deprecated Protocol Negotiation. Added Function Blocks. Many other notable changes, new features, and editorial improvements.

Contents

Figures

Tables

1 Introduction

1.1 Executive Summary

MIDI is a longstanding and entrenched specification. The features of MIDI 1.0 continue to work well after many years. 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.

MIDI 2.0 expands the feature set and capabilities of MIDI.

There are several key hurdles and requirements to consider as we make any additions to MIDI:

- Backwards compatibility is a key requirement. Users expect new MIDI Devices to work seamlessly with MIDI Devices sold in previous decades.
- All MIDI Status Bytes in MIDI 1.0 are defined. The opcodes and data payloads are defined. It is difficult to add new mechanisms, define any new messages or change the format of the existing 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 compatible messages to 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 which allows expansion of 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 includes queries for three major areas of expanded MIDI functionality:

1. Profile Configuration
2. Property Exchange
3. Process Inquiry

1.2 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. If a Device does not support new features, MIDI continues to work as defined by MIDI 1.0. Goals of MIDI-CI design include:

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 using Profiles.
6. Define method for Discovering, Getting, and Setting a wide range of Device Properties.
7. Define method for retrieving information about the current MIDI Message values from a Device.

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.

Even devices that do not implement any Category of MIDI-CI negotiations (Profile Configuration, Property Exchange, or Process Inquiry) are encouraged to use MIDI-CI Discovery. The central feature of the minimum requirements is the ability to Send and Respond to MIDI-CI Discovery messages.

1.3 Changes in this Version of MIDI-CI

This version contains all errata and feature changes or additions since MIDI-CI version 1.1 ***[MA03]***. The most significant changes include:

- Added features for backward and forward compatibility through future update versions. To ensure compatibility with previous and future MIDI-CI versions, every MIDI-CI implementation shall follow the rules defined in Section 5.4.
- Deprecated Protocol Negotiation.

  These mechanisms have been replaced by UMP Endpoint mechanisms defined in the M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol, Version 1.1 specification ***[MA06].***

    - Removed Bidirectional Settings and Authority Level sections (were only used by Protocol Negotiation).
- Added Section 9, Process Inquiry. This version of MIDI-CI introduces a new category of MIDI-CI, Process Inquiry, which allows an Initiator to discover the current state of supported MIDI Messages including:
    - System Messages
    - Channel Controller Messages
    - Note Data Messages
- Added Section 7.6, Profile Details Inquiry Message
  The Profile Details Inquiry message is used by an Initiator to discover details of the Profile implementation of a Responder. For example, an Initiator might discover:
    - How many Channels may be used or assigned by the Responder for a multi-channel Profile.
    - The Note Range supported by the Responder for the Profile.
    - Which optional Profile features the Responder is capable of supporting.

  The details which may be discovered are defined by the specification for the requested Profile Id.
- Added Profile Added Report and Profile Removed Report List messages.
- Added Section 3.1, Function Blocks.
  A Device may have one or more Function Blocks. A Function Block is a single functional component or application that operates on a set of one or more Groups. Function Blocks are used to separate a Device's functional components into manageable units.
- Expanded addressing capabilities due to the addition of Function Blocks.
- Fixed an error in the SubID#2 for the Invalidate MUID message in MIDI-CI Version 1.1 ***[MA03]***. The specification defined a SubID#2 with value 0x7E, while the table in the final Appendix listed it as 0x72. The value 0x7E is the correct value.

1.4 References

1.4.1 Normative References

[COMM01] ***CommonMark Spec***, Version 0.28, https://spec.commonmark.org/0.28/

[ECMA01] ***The JSON Data Interchange Syntax***, ECMA-404, https://www.ecma-international.org/publications/standards/Ecma-404.htm

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

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

[MA03] ***M2-101-UM MIDI Capability Inquiry (MIDI-CI)***, Version 1.1, the previous version of this specification, Association of Musical Electronics Industry, http://www.amei.or.jp/, and The MIDI Association, https://www.midi.org/

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

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

[MA06] ***M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol***, Version 1.1, Association of Musical Electronics Industry, http://www.amei.or.jp/, and The MIDI Association, https://www.midi.org/

[MA07] ***M2-113-UM MIDI-CI Profile: Default Control Change Mapping***, Version 1.0, Association of Musical Electronics Industry, http://www.amei.or.jp/, and The MIDI Association, https://www.midi.org/

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

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

1.5 Terminology

1.5.1 Definitions

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

Category: A collection of MIDI-CI messages grouped together because they function together or address similar types of mechanisms.

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

Data Set: A complete Property Exchange message whether sent in one System Exclusive message in single Chunk or in multiple Chunks.

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

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

Device ID: A one-byte field in Universal System Exclusive messages, as defined in the MIDI 1.0 Specification ***[MA01],*** to indicate which device in the system is supposed to respond. The more specific application of Device ID in MIDI-CI messages is defined in Section 3.2. The use of "Device" in this context is not the same as a Device as defined in this specification.

Function Block: A single logical entity which describes the functional components available on a UMP Endpoint of a Device. A Function Block operates on a set of one or more Groups. See the M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification ***[MA06]***.

Group: A field in the UMP Format addressing some UMP Format MIDI messages (and some UMPs comprising any given MIDI message) to one of 16 Groups. See the M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification ***[MA06]***.

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. The role of Initiator and Responder may alternate between the two MIDI-CI Devices. Either MIDI-CI Device may initiate a MIDI Transaction (act as Initiator) at any time. Also see Responder.

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

JSON: JavaScript Object Notation as defined in ***[ECMA01]***.

MA: See MIDI Association.

Message Format Version: The version reported in the Message Format Version field of the Discovery message.

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

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

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

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

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

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

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-CI Transaction: A Transaction using a set of MIDI-CI messages that includes an Inquiry sent by an Initiator 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. See also Transaction.

MIDI Endpoint: A Device which is an original source of MIDI messages or final consumer of MIDI messages.

MIDI In: A hardware or software MIDI connection used by a MIDI Device to receive MIDI messages from a MIDI Transport.

MIDI Out: A hardware or software MIDI connection used by a MIDI Device to transmit MIDI messages to a MIDI Transport.

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

MIDI Port: A hardware or software connector associated with a MIDI Endpoint using messages in MIDI 1.0 data format.

MIDI Thru: A hardware or software MIDI connection used by a MIDI Device to retransmit MIDI messages the device has received from a MIDI In.

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

MMA: See MIDI Manufacturers Association.

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

Note Data Messages: MIDI messages which include a Note Number field. These include Note On/off, Poly Pressure, Per-Note Pitchbend, Registered Per-Note Controllers, Assignable Per-Note Controllers, and Per-Note Management Message**.**

PE: Property Exchange.

Port: See MIDI Port.

Process Inquiry: A set of MIDI-CI Transactions by which one Device may discover the current state of supported MIDI Messages in another device.

Profile: An MA/AMEI specification that includes a set of MIDI messages and defined responses to those messages. A Profile is controlled by MIDI-CI Profile Negotiation Transactions. A Profile may have a defined minimum set of mandatory messages and features, along with some optional or recommended messages and features. See the MIDI-CI specification ***[MA03]*** and the Common Rules for MIDI-CI Profiles ***[MA04]***.

Property: A JSON key-value pair used by Property Exchange, for example `"channel": 1`.

Property Exchange: A set of MIDI-CI Transactions by which one device may access Property Data from another device.

Property Key: The key in a JSON key-value pair used by Property Exchange.

Property Value: The value in a JSON key-value pair used by Property Exchange.

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

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

Resource: A defined collection of one or more PE Properties with an associated inquiry to access its Properties.

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 the Initiator Device. Also see Initiator.

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

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

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

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

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

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

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

Reserved Words and Specification Conformance

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

Table 2 Words Relating to Specification Conformance

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

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

Table 3 Words Not Relating to Specification Conformance

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

1.6 Data Formats

Bitmap fields in MIDI-CI messages are presented as follows:

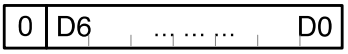


Figure 1 MIDI-CI Category Supported Bitmap

A single byte bitmap has data bits 0 through 6.

The most significant bit of byte is always the Status bit. This is always set to zero. (See the Complete MIDI 1.0 Detailed Specification ***[MA01]***).

2 Topology

2.1 Bidirectional

MIDI-CI requires bidirectional MIDI communications.

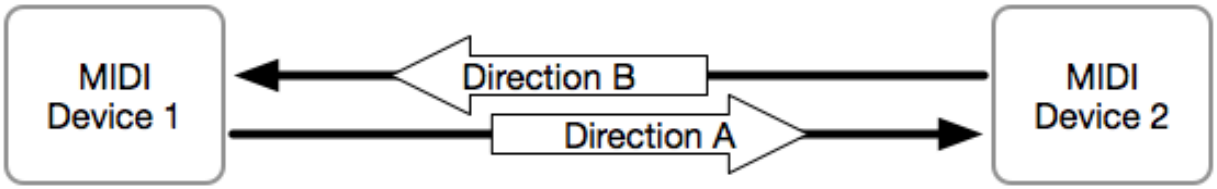


Figure 2 Bidirectional Communication

Every MIDI-CI capable Input shall be paired with a matching Output. Devices may have multiple pairs of MIDI Inputs and Outputs.

Each pair of 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 Sender "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 Sender can also ask a Receiver 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 for a specific MIDI-CI Transaction with a bidirectional connection.

Either of the two Devices may choose to function as the Initiator for a MIDI-CI Transaction.

- 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 Sender to Responder as MIDI Receiver.
- If a Device which has been acting as a Responder also wants to set parameters for communication, then the Responder may act as an Initiator for subsequent MIDI-CI Transactions to manage the configuration of the two Devices.

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 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.

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
    - Any Function Blocks discovered in establishing the connection
2. The UMP Format Group Field
    - Any Function Blocks which have the Group as a member
3. System Exclusive Device ID Fields
4. MIDI-CI Devices' MUIDs

3.1 Function Blocks

A Device may have one or more Function Blocks. A Function Block is a single functional component or application that operates on a set of one or more Groups. Function Blocks are used to separate a Device's functional components into manageable units. See the M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification ***[MA06]***.

Each Function Block that supports MIDI-CI shall have a different MUID and act as a unique MIDI-CI Device (See Section ***3.3.1***).

MIDI-CI Messages that are meant for a specific Function Block, Group in a Function Block, or Channel in a Group, are sent using the UMP Group Field that the MIDI-CI Message is addressing. (For more details of addressing MIDI-CI messages, See Section ***5.2.1***.)

If a UMP Endpoint does not declare any Function Block topology, then the user has less information on how to configure the Devices to be on matching Group (and Channel) addresses for communication.

MIDI-CI Devices that use the MIDI 1.0 data format do not use the UMP Format and therefore will not have Groups or Function Blocks.

3.1.1 USB MIDI 2.0 Group Terminal Blocks

The USB MIDI 2.0 Specification version 1.0 [USBIF02] describes the use of Group Terminal Blocks to declare the entities that operate on a set of one or more Groups. MIDI-CI treats Group Terminal Blocks like Function Blocks. Each Group Terminal Block should have its own MUID. Also see the M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification ***[MA06]***.

The USB MIDI 2.0 specification describes the bMIDIProtocol value 0x00 as "Unknown (Use MIDI-CI)". However, the Protocol Negotiation mechanisms in MIDI-CI have been deprecated. MIDI-CI Protocol Negotiation has been replaced by Endpoint Discovery, Endpoint Info Notification, Stream Configuration Request, and Stream Configuration Notification mechanisms defined in the M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification ***[MA06]***.

3.2 MIDI-CI Addressing

MIDI messages sent in the MIDI 1.0 data format can be addressed to any one of sixteen Channels (for example Note On/Off, controllers, etc.) or to the whole MIDI Endpoint (System messages).

MIDI 2.0 and the Universal MIDI Packet Format expand on the available addresses, with sixteen Channels in each of sixteen Groups. These sixteen Groups are addressed on a single UMP Endpoint. One or more Groups which are used together for a common function are members of a Function Block.

MIDI-CI provides mechanisms for all these addresses.

3.2.1 MIDI-CI Addressing in MIDI 1.0 Data Format Connections

The Device ID field of Universal System Exclusive MIDI-CI Messages is used for addressing:

- 0x00-0x0F map to MIDI Channels 1-16
- 0x7E is addressed to the whole MIDI Endpoint, not channelized.
- 0x7F is addressed to the whole MIDI Endpoint, not channelized, and treated as a Function Block with only one member Group.

Whether the Device ID refers to the Source or Destination depends on the definition of each message type. See Section ***5.2.1*** and individual messages for more details.

3.2.2 MIDI-CI Addressing in UMP Format Connections

MIDI-CI messages in the UMP Format are addressed according to the combination of two fields in the message: the UMP Group field and the Universal System Exclusive Device ID field.

The UMP Group field identifies:

- A specific Group

Values in the Device ID field of Universal System Exclusive MIDI-CI Messages are used for further addressing:

- 0x00-0x0F map to MIDI Channels 1-16 in the UMP Group
- 0x7E addresses to the UMP Group (not channelized)
- 0x7F addresses any Function Block which has the UMP Group as a member, or if there are no Function Blocks declared then to the UMP Group and treated as a Function Block with only one member Group.

Whether the Device ID refers to the Source or Destination depends on the definition of each message type. See Section ***5.2.1*** and individual messages for more details.

3.3 MIDI-CI MUID

MUID is a 28-bit random number generated by a MIDI-CI Device used to uniquely identify MIDI-CI messages to or from a Function Block or other functional entity of that Device. All MIDI-CI messages include the MUID of the source Function Block and the MUID of the destination Function Block. For more information about Function Blocks see Section ***3.1***.

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 ***3.3.3***).

MIDI-CI provides a mechanism to associate a MUID with a Function Block (See Section ***5.6.2***). For more information about Function Blocks see Section ***3.1***.

3.3.1 Generating a MUID

Every time a MIDI-CI Device is powered up, it shall create a new, randomly generated MUID for each Function Block. 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 on that Function Block (see Section ***5.6.1***) for that MUID. When a Device is restarted it shall generate new MUIDs for each Function Block.

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

If a MUID for a Function Block is changed (invalidated and a new MUID is generated) the Device shall send a Function Block Info Change Alert UMP Message (See the M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification ***[MA06]***).

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 A***).

3.3.2 Invalidating a MUID

MUIDs may be invalidated for several reasons, including but not limited to:

- MIDI-CI Device is shutting down or rebooting.
- MUID Collision – In the rare case that a collision does occur, MIDI-CI provides rules and mechanisms for resolving the collision (see Sections ***5.9.1*** and ***5.9.2***).
- If a Function Block or other functional entity of a MIDI-CI Device changes its Group location or number of Groups, the Device shall invalidate the MUID (See Section ***5.9***). After the change the Device shall generate a new MUID and establish a new MIDI-CI connection (See Section ***4***).

3.3.3 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 Receivers.

Broadcast MUID = 0x0FFFFFFF (expressed as 4 bytes of 7-bit data in MIDI-CI message: 0x7F 0x7F 0x7F 0x7F)

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 Added Report
- Profile Removed Report
- 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.

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 should 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 Initiator's MUID as the destination.

4.2 Subsequent Transactions

Initiator and Responder roles are not fixed. Roles are per Transaction and only persist until that Transaction is completed.

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.

Table 4 Categories of MIDI-CI Messages

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

Note: Sub-ID#2 Range 0x10-0x1F was defined for Protocol Negotiation in previous versions of MIDI-CI **[MA03]. Protocol Negotiation is deprecated as of this version 1.2 of MIDI-CI.*

5.2 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. When MIDI-CI messages are 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.

5.2.1 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.

Table 5 Standard Format for MIDI-CI Messages

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Device ID: Source or Destination (depending on type of message): 00–0F: To/from MIDI Channels 1-16 10–7D: Reserved 7E: To/from Group 7F: To/from Function Block

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 (Deprecated) 0x20–2F: Profile Configuration Messages 0x30–3F: Property Exchange Messages 0x40-4F: Process Inquiry 0x50–6F: Reserved 0x70–7F: Management Messages
1 byte	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first*)
4 bytes	Destination MUID (LSB first*)
n 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 0x7E are addressed to the whole Group, not channelized. Messages with a Value of 0x7F are addressed to the whole Function Block, not channelized.

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

In a MIDI-CI inquiry message sent by the Initiator, this is the destination of the inquiry. Default value is 0x7F = Function Block based inquiry. But the value can be 0x00-0x0F to address a specific MIDI Channel, or 0x7E to address a specific Group.

In a MIDI-CI reply message sent by the Responder, this is the source of the reply. Default value is 0x7F = Function Block based reply. But the value can be 0x00-0x0F to reply about a specific MIDI Channel, or 0x7E to address a specific Group.

Values 0x00-0x0F are for 16 MIDI channels. This allows inquiries and negotiation on a specific channel. This is useful for using Profile Configuration on a per channel basis. Values 0x00-0x0F and 0x7E shall not be used for Property Exchange messages.

If a Device sends a message using MIDI-CI Device ID = 0x7F, the message may be sent using any UMP Group field value which is within the range of the Function Block. When a Receiver receives a message with MIDI-CI Device ID = 0x7F, it shall send any related replies using the same UMP Group field value.

1 byte MIDI-CI Message Version/Format

In version 1.1 of the MIDI-CI specification, the version number is 0x01. In this version 1.2 of the MIDI-CI Specification, the version number is 0x02.

The Message Format Version is broken into 2 parts:

- bits 7..5 – Reserved for future major revisions
- bits 4..1 – Minor Version number

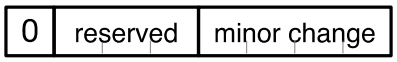


Figure 3 MIDI-CI Message Version/Format Field

If any of the reserved bits are set, then that version may have features which are incompatible with the format of messages in this version.

See Section *5.3* For more information about using the MIDI-CI Version/Format field.

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.

5.3 Handling the MIDI-CI Message Version/Format Field

When sending MIDI-CI messages, a Device shall always use its own Message Format Version, which might not be the same as the Device it is communicating with. Senders should avoid sending messages introduced in a later Message Format Version than the Receiver supports. If a Device wishes to change to sending a different Message Format Version, the Device shall invalidate its MUID and initiate a new Discovery Transaction.

When receiving a MIDI-CI message, the Device shall examine the received Message Format Version to determine response behavior.

- If the received message is of the same version as the Device, then no special considerations are necessary.
- If the received version is higher than the Device's supported version, the Device shall only process the fields defined for its supported version and ignores any appended fields, as well as any reserved values and bits.
- If the received version is lower than the Device's supported version, the Device shall only process the fields, values, and bits defined in the received version of MIDI-CI.
- If any of the reserved bits of the Message Format Version field are set, the Device shall reply with a NAK message (error code 0x02, see section *5.11*).

5.4 MIDI-CI Versions and Backward Compatibility

In order to ensure compatibility with previous and future MIDI-CI versions, every MIDI-CI implementation shall follow these rules:

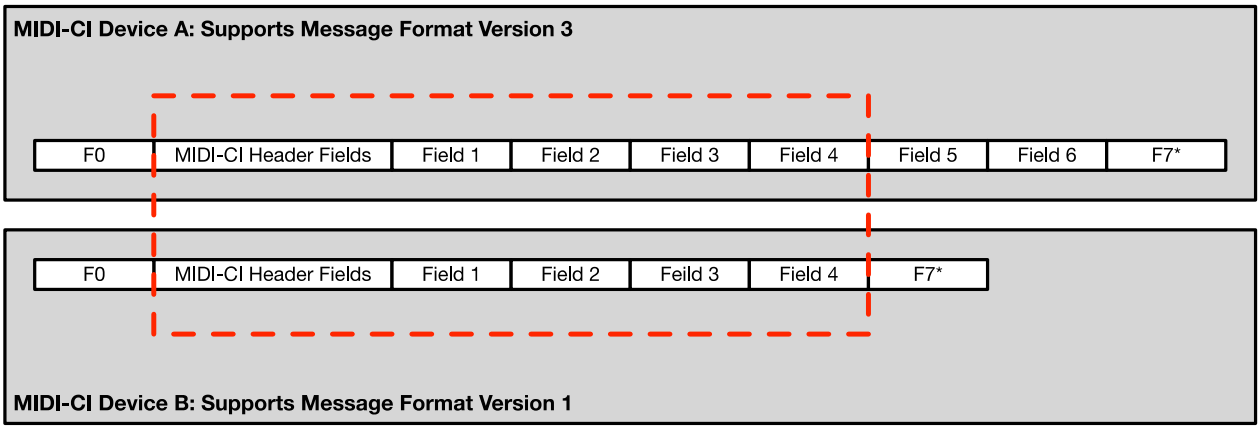
- Devices shall use Message Format Version 0x01 or higher. Message Format Version 0x00 is deprecated and shall not be used.
- In addition to the given Message Format Version that a Device uses, the device shall be able to receive and parse all previous valid versions for all MIDI-CI functions supported by the Device. This rule applies even if Inquiries from previous versions have been deprecated in later versions of MIDI-CI.
  Example: A Device that uses Message Format Version 0x03 shall also support reception of versions 0x02 and 0x01.
- MIDI-CI Devices are not required to send messages in earlier Message Format Versions.
- When a Device receives messages of a higher Message Format Version than it implements, it should use the Message fields that are known, and ignore any unknown Message fields received.

- When a Device receives messages of a lower Message Format Version it shall process all relevant Message fields received.
- For sending, devices shall not mix Message Format versions. Devices shall send all messages in the same Message Format Version reported in Discovery and Reply to Discovery.
- If a Device needs to change to a different version of MIDI-CI, it shall invalidate its MUID and initiate a new Discovery Transaction with the new version.
- A Receiver of a Discovery message (or any other message) with the reserved version bits set, reply with a NAK message using the Receiver's Message Format Version and NAK Status Code set to 0x02.

Future MIDI-CI messages in the minor version updates shall only add fields to the end of a message (before F7) – never alter a field (excluding reserved bytes/bits). Major version updates may structurally change and remove MIDI-CI messages.

This means that MIDI-CI implementations can handle unsupported minor version updates by ignoring all unknown extended fields.

5.4.1 Examples of Handling Different Versions of MIDI-CI



*Note: F0 and F7 Only When Not Using UMP

Figure 4 Two Devices with Different Version/Format of MIDI-CI

Example 1:

Device A sends to Device B

When Device B receives a MIDI-CI message from Device A, it shall process all the MIDI-CI Header fields and Properties 1 through 4. Device B shall ignore Properties 5 and 6.

Example 2:

Device B sends to Device A

When Device A receives a MIDI-CI message from Device B, it shall process all the MIDI-CI Header fields and Properties 1 through 4.

Example 3:

Device A sends an Inquiry message to Device B. Device A knows that Device B only supports v1, but it must still use the v3 format of the Inquiry message. The additional fields it assembles and sends in the Inquiry will be ignored by Device B. Device B will reply with the v1 response.

Example 4:

Device B sends an Inquiry message to Device A in format v1. Device A must respond with the v3 format of the message, well knowing that Device B will ignore all fields that are new in v2 and v3 of that message. Device B will receive the v3 message, ignoring the additional fields it does not understand.

5.5 Discovery Message

An Initiator shall establish connections to MIDI-CI Responders 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.
- A MIDI-CI capable Device should send a Discovery message when the Device or a Function Block in the Device has had its previous MUID invalidated and wants to re-establish MIDI-CI connections using a new MUID.

Table 6 MIDI-CI Discovery Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	Device ID: 7F = To/from Function Block
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)
The following fields (except F7 End) were added in MIDI-CI Message Version 2	
1 byte	Initiator's Output Path Id
F7	End Universal System Exclusive

5.5.1 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 ***[MA01]***). 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 the version of software or firmware but may also be the 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.

5.5.2 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 is not required to declare support.

Table 7 MIDI-CI Category Supported Bitmap Bit Allocation

Bit	Category	Supported Sub-ID# Range	Description
D0	0	0x00-0x0F	Reserved – No Messages Defined Yet
D1	1	0x10-0x1F	Protocol Negotiation (Deprecated)
D2	2	0x20-0x2F	**Profile Configuration Supported**
D3	3	0x30-0x3F	**Property Exchange Supported**
D4	4	0x40-0x4F	**Process Inquiry Supported**
D5	5	0x50-0x5F	Reserved – No Messages Defined Yet
D6	6	0x60-0x6F	Reserved – No Messages Defined Yet

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

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.*

5.5.3 Receivable Maximum SysEx Message Size

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.

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.

5.5.4 Output Path Id

Initiators that have multiple MIDI Out connections should use a unique Output Path Id for each connection (even on the same MUID), to enumerate/differentiate each output. Each MIDI Out connection should use an unchanging Output Path Id number.

The Initiator might number its MIDI Outputs starting with 0 and sequentially incrementing the value for each additional MIDI Output. This allows up to 128 unique MIDI Outputs. Initiators with only one MIDI Out connection or which do not need this feature, should set this to 0.

5.5.5 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.

Table 8 Reply to Discovery Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	Device ID: 7F = from Function Block
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)*
The following fields (except F7 End) were added in MIDI-CI Message Version 2	
1 byte	Initiator's Output Path Instance Id (from the Discovery message received)
1 byte	Function Block

	Set to 0x7F if no Function Block
F7	End Universal System Exclusive

*See the Receivable Maximum SysEx Message Size in Section 5.5.3.

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 this reply to inform the Initiator of the existence of a device which does not support any MIDI-CI functions.

5.6.1 Output Path Id

The Reply to Discovery shall return the same Output Path Id provided in the originating Discovery Message.

5.6.2 Function Block

The Reply to Discovery shall declare the number of the Function Block that this Responder represents. This allows an Initiator to tie a Function Block to a MIDI-CI Device. This allows the Initiator to know the UMP Groups associated with this MIDI-CI Device. If the MIDI-CI Device is not associated with a Function Block (for example, is connected by a MIDI 1.0 byte stream), then set the value to 0x7F (no Function Block). See Section ***3.1*** for more information about Function Blocks.

5.7 Inquiry: Endpoint Message

An Initiator may send the Inquiry: Endpoint Message to a Function Block in a Responder to get information about the UMP Endpoint which has the Function Block. A Status field selects the target data.

Table 9 Inquiry: Inquiry: Endpoint Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	Device ID: 7F = from Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
72	Universal System Exclusive Sub-ID#2: Inquiry: Endpoint Information
1 byte	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
1 byte	Status
F7	End Universal System Exclusive

5.7.1 Status

The Status field defines which information to retrieve from the Responder.

Table 10 Inquiry: Endpoint Information Status Values

Status Value	Target Property	Details
0x00	Product instance ID	Product Instance ID of the UMP

		Endpoint to which the Responder belongs
0x01-0x7F	Reserved	

In this version of MIDI-CI, the only defined Status value is 0x00 which is used to get the Product Instance Id of the UMP Endpoint which has the Function Block addressed by the inquiry. This inquiry gets the Product Instance Id which is also declared in a Product Instance Id Notification message (See the M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol, Version 1.1 ***[MA06]***).

5.8 Reply to Endpoint Message

When a MIDI-CI Device receives an Inquiry: Endpoint Information message, it should reply with the Reply to Endpoint Information message. If the Device is not able to provide the requested information of the given Status, it should reply with a MIDI-CI NAK message (See Section ***5.11***).

Table 11 Reply to Endpoint Information Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	Device ID: 7F = from Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
73	Universal System Exclusive Sub-ID#2: Reply to Endpoint Information
1 byte	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
1 byte	Status
2 bytes	Length of Following Information Data (lid) (LSB first)
lid bytes	Information Data
F7	End Universal System Exclusive

5.8.1 Status

The Responder sets the Status field to the same value it received in the Inquiry: Endpoint Information message. The Status value determines the type of information that follows in the Information Data field.

5.8.2 Length of Following Information Data

This field specifies the length in bytes of the following Information Data field.

5.8.3 Information Data

This variable-length field contains the data requested, depending on the Status value.

5.8.3.1 Information Data for Status Value 0x00: Endpoint Product Instance Id

When sending the Reply to Endpoint Information with Status 0x00, the Information Data field contains the Endpoint Product Instance Id. The value shall match the Product Instance Id which is declared in a Product Instance Id Notification message (See the M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol, Version 1.1, ***[MA06]***).

Product Instance Id should be, where possible, the same as the Serial Number of the Device and should be a unique number per Manufacturer/Family/Model. The Product Instance Id should be persistent across power cycles.

Product Instance Id shall be ASCII Text in the ordinal range 32-126.

The Product Instance Id shall not be any longer than 16 bytes in size,

The Product Instance Id can be used to:

- Re-identify a Device after power cycling
- Re-identify a Device after a MUID invalidation
- Identify the same Device if using multiple Function Blocks each with its own MUID
- Distinguish multiple Devices of the same model
- Facilitating MUID collision handling

5.9 Invalidate MUID Message

An Invalidate MUID message is used as defined in Section ***3.3.2***.

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.

Table 12 MIDI-CI Invalidate MUID Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	Device ID: 7F = to Function Block
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. To re-establish MIDI-CI connections using a new MUID, the Device which has had its MUID invalidated should send a new MIDI-CI Discovery Message to all connections.

5.9.1 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

5.9.2 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

5.10 MIDI-CI ACK Message

The MIDI-CI ACK Message is a message for dealing with positive acknowledgement of an action, or to provide a notice of ongoing activity, such as timeout wait messages.

Table 13 MIDI-CI ACK Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Device ID: Source 00–0F: To/from MIDI Channels 1-16 10–7D: Reserved 7E: To/from Group 7F: To/from Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
7D	Universal System Exclusive Sub-ID#2: MIDI-CI ACK

1 byte	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
1 byte	Original Transaction Sub-ID#2 Classification e.g. 0x34
1 byte	ACK Status Code
1byte	ACK Status Data
5 bytes	ACK details for each Sub Id Classification
2 bytes	Message Length (ml) (LSB first)
ml bytes	Message Text
F7	End Universal System Exclusive

Note: This MIDI-CI message is defined for new mechanisms which are not published at the time of this specification. Look for usage explanations in other specifications which reference this message.

5.10.1 Original Transaction Sub-ID#2 Classification

A MIDI-CI ACK message declares the result of a successful transaction or declares the status of an in-progress MIDI-CI transaction. The Original Transaction Sub-Id#2 Classification is used, along with other fields, to link the ACK message with the original Transaction.

5.10.2 ACK Status codes and ACK Status Data

The ACK Status code provides more information as to the cause of the issue. Some Status Codes require additional information in the Status Data field to help process the ACK. Future documents will describe other ACK Status Codes and what they declare. If no additional information is needed, set the value to 0x00.

Table 14 MIDI-CI ACK Status Codes

Status Code	Status Data	Reason
0x00 - 0x0F Success Messages – Do Not Retry		
0x00	0x00	ACK
0x10 - 0x1F Notification Messages – These require further action		
0x10	0x00-7F	Timeout Wait. Status Data is the time to wait in multiples of 100 milliseconds (0-12.7 seconds).

5.10.3 ACK details for each Sub Id Classification

Each branch of MIDI-CI requires unique information that relates to the branch. The 5 bytes are used to describe details about the MIDI-CI Branch used:

- Profile Configuration - the 5 bytes used here represent the Profile Id
- Property Exchange - Byte 1 is the PE Stream Id. Byte 2 and 3 represent the current Chunk number (LSB first). Other bytes are reserved.
- Process Inquiry - The 5 Bytes are reserved.

5.10.4 Message Length (ml) and Message Text Encoding

Messages provide extended human readable text that further explains the reason for the MIDI-CI ACK message. These messages should be presented to the user.

ACK messages shall not exceed the responder's Maximum Receivable SysEx Size as established by MIDI-CI Discovery. Therefore, it is generally safe to send up to 103 bytes of Message text.

7-Bit Encoding of Message Text

Due to the restrictions of MIDI 1.0 SysEx, Message values have the following rules:

1. 7-bit ASCII characters are supported, except that the "\" character has a special function as defined in the JSON standard, ECMA404 [EXT01].

   For clarity in MIDI-CI, as ECMA404 may be unclear about this, the forward slash character "/" can be optionally escaped so it can be encoded in Message values as either "/" or "\/".

2. All characters which are not 7-bit ASCII characters shall be converted to UTF-16, and then escaped using "\u" as defined in the JSON standard, ECMA404 [EXT01].

   Note: These are pure ASCII characters. Therefore, the UTF-8 rules defined in the UMP and MIDI 2.0 Protocol specification **[MA06]** *do not apply.*

3. The Message should not use ASCII Characters below 0x20 (space), except for 0x0A (line feed). Other ASCII Characters below 0x20 such as 0x08 (tab) and 0x0D (carriage return) should be ignored.

Unicode Characters Examples:

1. Accepted Beat ♪ is converted to Accepted Beat \u266a.
2. 残りわずか5バイト is converted to \u6B8B\u308A\u308F\u305A\u304B5\u30D0\u30A4\u30C8

5.11 The MIDI-CI NAK 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.
- Reply to a MIDI-CI message with a MIDI-CI Message Format Version with reserved bits set.

Table 15 MIDI-CI NAK Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Device ID: Source 00–0F: To/from MIDI Channels 1-16 10–7D: Reserved 7E: To/from Group 7F: To/from Function Block
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)
The following fields (except F7 End) were added in MIDI-CI Message Version 2	
1 byte	Original Transaction Sub-ID#2 Classification e.g. 0x34
1 byte	NAK Status Code
1byte	NAK Status Data
5 bytes	NAK details for each Sub Id Classification
2 bytes	Message Length (ml) (LSB first)
ml bytes	Message Text
F7	End Universal System Exclusive

5.11.1 Original Transaction Sub-ID#2 Classification

A MIDI-CI NAK message is always the result of a failed MIDI-CI transaction. The Original Transaction Sub-Id#2 Classification is used, along with other fields, to link the NAK message with the original Transaction.

5.11.2 NAK Status codes and NAK Status Data

The NAK Status code provides more information as to the cause of the issue. Some Status Codes require additional information in the Status Data field to help process the NAK. If no additional information is needed, set the value to 0x00.

Table 16 MIDI-CI NAK Status Codes

Status Code	Status Data	Reason
0x00 - 0x1F Fail Messages – Do Not Retry		
0x00	0x00	NAK
0x01	0x00	MIDI-CI message not supported
0x02	0x00	MIDI-CI version not supported
0x03	0x00	Channel/Group/Function Block Not in use
0x04	0x00	Profile not supported on the requested Channel, Group, or Function Block (See also Common Rules for Profiles ***[MA04]***)
0x20 - 0x3F Notification Messages – further action is recommended		
0x20	0x00	Terminate Inquiry (replaces PE Notify Message with a status of 144)
0x21	0x00	Property Exchange Chunks are out of sequence (cannot recover)
0x40 - 0x5F Fail Message – Retry is recommended		
0x40	0x00	Error occurred, please retry
0x41	0x00	Message was malformed

0x42	0x00	Timeout has occurred
0x43	0x00-0x7F	Busy, try again time, in multiples of 100 milliseconds (0-12.7 seconds)
0x60 – 0x7F Reserved		

Note: A Device receiving a NAK Messages with a status code greater than 0x0F may decide to not retry or perform additional actions and may choose to halt the process instead.

5.11.3 NAK details for each Sub Id Classification

Each branch of MIDI-CI requires unique information that relates to the branch. The 5 bytes are used to describe details about the MIDI-CI Branch used:

- Profile Configuration - the 5 bytes used here represent the Profile Id
- Property Exchange - Byte 1 is the PE Stream Id. Byte 2 and 3 represent the current Chunk number (LSB first). Other bytes are reserved.
- Process Inquiry - The 5 Bytes are reserved.

5.11.4 Message Length (ml) and Message Text Encoding

Messages provide extended human readable text that further explains the reason for the MIDI-CI NAK message. These messages should be presented to the user.

Message Length is in bytes. NAK messages shall not exceed the responder's Maximum Receivable SysEx Size as established by MIDI-CI Discovery. Therefore, it is generally safe to send up to 103 bytes of Message text.

Message Text uses the same format as described in Section 5.10.4.

6 Protocol Negotiation

Version 1.1 of MIDI-CI included mechanisms for two Devices to negotiate which Protocol to use for communication. Those mechanisms are deprecated beginning with version 1.2 of MIDI-CI. They have been replaced by UMP Endpoint mechanisms defined in the M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol, Version 1.1 specification ***[MA06]***.

For definitions of Protocol Negotiation mechanisms and the deprecated messages, see MIDI-CI version 1.1 ***[MA03]***.

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 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.

The MIDI-CI specification contains only the base message definitions used for Profile Configuration. Critical information and further rules for implementing Profile Configuration are defined in the M2-102-U Common Rules for MIDI-CI Profiles specification ***[MA04]***.

Detailed information and further implementation rules for individual Profiles are defined in specifications for each Profile.

All Devices which implement a Profile shall comply with the rules of this Section 7 Profile Configuration, the rules of theM2-102-U Common Rules for MIDI-CI Profiles ***[MA04]*** specification, and the rules of the specification for the Profile.

7.1 Profile Configuration Mechanism

Profiles are controlled by the following Common Profile Configuration messages:

- Profile Inquiry
- Reply to Profile Inquiry
- Profile Added Report
- Profile Removed Report
- Inquiry: Profile Details Message
- Reply to Profile Details Message
- Set Profile On
- Set Profile Off
- Profile Enabled Report
- Profile Disabled Report

These messages are defined in this specification but details of implementing the messages are further defined in the Common Rules for MIDI-CI Profiles specification ***[MA04]***.

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

7.2 Profile Inquiry Message

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

Table 17 Profile Inquiry Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Destination 00–0F: To/from MIDI Channels 1-16 10–7D: Reserved 7E: To/from Group 7F: To/from Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI

20	Universal System Exclusive Sub-ID#2: Profile Inquiry
1 byte	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 two 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 18 Reply to Profile Inquiry Message Format

<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
00–0F: To/from MIDI Channels 1-16
10–7D: Reserved
7E: To/from Group
7F: To/from Function Block</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>1 byte</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: (Optional)
<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>
</table>

(cep - 1) x 5 bytes	Optional: Profile ID of Other Currently Enabled Profiles in sets of 5 bytes.
	…
	Optional: Profile ID of Last Currently Enabled Profile
2 bytes	Number of Currently Disabled Profiles Supported (cdp) (LSB first)
5 bytes	Profile ID of First Currently Disabled Profile Supported (Optional)
(cdp - 1) x 5 bytes	Optional: Profile ID of Other Currently Disabled Profiles Supported in sets of 5 bytes.
	…
	Optional: Profile ID of Last Currently Disabled Profile Supported
F7	End Universal System Exclusive

7.3.1 Profile ID

Each Profile has a 5-byte identifier. Standard Defined Profiles are those adopted by AMEI and MA. The value of the Profile ID Byte 1 is 0x7E (Universal). Bytes 2-4 for each Profile uses the ID values defined by the Profile Specification and by other AMEI/MA Profile related specifications.

Manufacturers may use MIDI-CI to control Manufacturer specific 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 two bytes are filled with zeroes.

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

Table 19 Five Byte Manufacturer Specific Profile ID Format

Byte Number	Standard Defined Profiles	Manufacturer Specific Profiles
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

7.3.2 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.

7.3.3 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.

7.4 Profile Added Report

While a Device is active, it might change its state such that a Profile is newly added to its capabilities. If the Device has previously acted as a Responder by sending a Reply to Profile Inquiry message, then the Device should send a Profile Added Report message to update the list of Currently Disabled Profile Supported declared in the previous Reply to Profile Inquiry message.

Table 20 Profile Added Report Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Device ID: Source or Destination (depending on type of message): 00–0F: To/from MIDI Channels 1-16 10–7D: Reserved 7E: To/from Group 7F: To/from Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
26	Universal System Exclusive Sub-ID#2: Profile Added Report
1 byte	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 Being Added
F7	End Universal System Exclusive

The Profile Added Report message shall have the Destination MUID set to Broadcast (0x7F 7F 7F 7F).

If the newly added Profile is enabled in the Device, then the Device shall then send a Profile Enabled Report message for that Profile immediately following the Profile Added Report.

7.5 Profile Removed Report

While a Device is active, it might change its state such that it is no longer capable of supporting a Profile. If the Device has previously acted as a Responder by sending a Reply to Profile Inquiry message, then the Device should send a Profile Removed Report message to update the list of Currently Disabled Profile Supported declared in the previous Reply to Profile Inquiry message.

Table 21 Profile Removed Report Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Device ID: Source or Destination (depending on type of message): 00–0F: To/from MIDI Channels 1-16 10–7D: Reserved

	7E: To/from Group 7F: To/from Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
27	Universal System Exclusive Sub-ID#2: Profile List Remove
1 byte	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 Being Removed
F7	End Universal System Exclusive

The Profile Removed Report message shall have the Destination MUID set to Broadcast (0x7F 7F 7F 7F).

If the Device is removing a Profile which is currently enabled, then the Device shall send a Profile Disabled Report message for that Profile before sending the Profile Removed Report message.

7.6 Profile Details Inquiry Message

The Profile Details Inquiry message is used by an Initiator to discover details of the Profile implementation of a Responder. For example, an Initiator might discover:

1. How many Channels may be used or assigned by the Responder for a multi-channel Profile.
2. Which optional Profile features the Responder is capable of supporting.

The details which may be discovered are defined by the specification for the requested Profile Id.

These details may be discovered before a Profile is enabled. For example, in some cases, it may be necessary to know the number of MIDI Channels available for use by a Profile before enabling that Profile. This message may also be used to discover details about a Responder's Profile implementation while it is enabled.

For more details see the M2-102-U Common Rules for MIDI-CI Profiles ***[MA04]*** specification.

Table 22 Profile Details Inquiry Message

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Destination 00–0F: To/from MIDI Channels 1-16 10–7D: Reserved 7E: To/from Group 7F: To/from Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
28	Universal System Exclusive Sub-ID#2: Profile Details Inquiry
1 byte	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)

4 bytes	Destination MUID (LSB first)
5 bytes	Profile ID of Profile
1 byte	Inquiry Target 0x00 - 0x3F - Registered Target Data 0x40 - 0x7F - Profile Specific Target Data
F7	End Universal System Exclusive

7.6.1 Inquiry Target:

This field declares the type of data that is being requested by the inquiry.

0x00 - 0x3F - Registered Target Data

These are specific data types as defined by MIDI Association / AMEI specifications including the M2-102-U Common Rules for MIDI-CI Profiles ***[MA04]*** specification. All Profiles shall have the same format of data for these Inquiry Targets.

0x40 - 0x7F - Profile Specific Target Data

These are data types and values defined by any Profile specification. Each Profile defines a unique format of data for these Inquiry Targets, so the meaning is Profile Specific.

7.7 Reply to Profile Details Inquiry Message

Table 23 Reply to Profile Details Inquiry Message

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Destination 00–0F: To/from MIDI Channels 1-16 10–7D: Reserved 7E: To/from Group 7F: To/from Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
29	Universal System Exclusive Sub-ID#2: Reply to Profile Details Inquiry
02	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
5 bytes	Profile ID of Profile
1 byte	Inquiry Target 0x00 - 0x3F - Registered Target Data 0x40 - 0x7F - Profile Specific Target Data
2 bytes	Inquiry Target Data Length (dl) (LSB first)

dl bytes	Inquiry Target Data.
F7	End Universal System Exclusive

7.7.1 Inquiry Target

The content of the Inquiry Target field shall be the same as in the Profile Details Inquiry message which was received.

7.7.2 Inquiry Target Data Length

This declares the number of bytes in the following Inquiry Target Data Field (two bytes, LSB first).

7.7.3 Inquiry Target Data

The Inquiry Target Data is the data from the Responder sent in response to the Inquiry Target which was requested by the Initiator.

If the Inquiry Target is 0x00 – 0x3F (Registered Target Data), then the format of the reply is defined in the M2-102-U Common Rules for MIDI-CI Profiles ***[MA04]*** specification.

If the Inquiry Target is 0x40 – 0x7F (Profile Specific Target Data), then the format of the reply is defined in the specification for the requested Profile Id.

7.8 Set Profile On Message

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

Table 24 Set Profile On Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Destination 00–0F: To/from MIDI Channels 1-16 10–7D: Reserved 7E: To/from Group 7F: To/from Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
22	Universal System Exclusive Sub-ID#2: Set Profile On
1 byte	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)
The following fields (except F7 End) were added in MIDI-CI Message Version 2	
2 bytes	Number Channels Requested (LSB First) to assign to this Profile when it is enabled*
F7	End Universal System Exclusive

*Note: When the Profile Destination field is set to address 0x7E or 0x7F, the number of Channels is determined by the width of the Group or Function Block. Set the Number of Channels Requested field to a value of 0x0000.

7.9 Set Profile Off Message

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

Table 25 Set Profile Off Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Destination 00–0F: To/from MIDI Channels 1-16 10–7D: Reserved 7E: To/from Group 7F: To/from Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
23	Universal System Exclusive Sub-ID#2: Set Profile Off
1 byte	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)
The following fields (except F7 End) were added in MIDI-CI Message Version 2	
2 bytes	Reserved
F7	End Universal System Exclusive

7.10 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 event enables a Profile. For example:

- This message should be sent if a user changes to a mode on a Device which results in enabling a Profile.
- This message should be sent if a Device receives a Program Change which results in enabling 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.

Table 26 Profile Enabled Report Message Format

Value	Parameter
F0	System Exclusive Start

7E	Universal System Exclusive
1 byte	Source 00–0F: To/from MIDI Channels 1-16 10–7D: Reserved 7E: To/from Group 7F: To/from Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
24	Universal System Exclusive Sub-ID#2: Profile Enabled Report
1 byte	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
The following fields (except F7 End) were added in MIDI-CI Message Version 2	
2 bytes	Number Channels enabled on this Profile. (Manager + Member Channels, LSB first)*
F7	End Universal System Exclusive

*Note: When the Profile Destination field is set to address 0x7E or 0x7F, the number of Channels is determined by the width of the Group or Function Block. In these cases, set the Number of Channels Enabled field to a value of 0x0000.

7.11 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 event disables a Profile. For example:

- This message should be sent if a user changes to a mode on a Device which results in disabling a Profile.
- This message should be sent if a Device receives a Program Change which results in disabling 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.

Table 27 Profile Disable Report Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Source 00–0F: To/from MIDI Channels 1-16 10–7D: Reserved 7E: To/from Group 7F: To/from Function Block

0D	Universal System Exclusive Sub-ID#1: MIDI-CI
25	Universal System Exclusive Sub-ID#2: Profile Disabled Report
1 byte	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
The following fields (except F7 End) were added in MIDI-CI Message Version 2	
2 bytes	Number Channels disabled on this Profile. (Manager + Member Channels, LSB first)*
F7	End Universal System Exclusive

*Note: When the Profile Destination field is set to address 0x7E or 0x7F, the number of Channels is determined by the width of the Group or Function Block. In these cases, set the Number of Channels Disabled field to a value of 0x0000.

7.12 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.

Table 28 Profile Specific Data Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Device ID: Source or Destination (depending on type of message): 00–0F: To/from MIDI Channels 1-16 7E: To/from Group 7F: To/from Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
2F	Universal System Exclusive Sub-ID#2: Profile Specific Data
1 byte	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.

7.13 Declaring No Currently Supported Profiles

If a Device does not support any Profiles, then it should declare that it does not support Profile Configuration in the Discovery Message (see Section *5.5*). If that Device receives a Profile Inquiry message, 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.

If a Device supports Profiles and it receives a Profile Inquiry message on a MIDI Channel, Group, or Function Block where the Device does not currently support any Profiles, the Device shall 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.

If a Device sometimes supports Profiles but is currently in a mode in which the Device does not support any Profiles and it receives a Profile Inquiry message, the Device should 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.

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.

The MIDI-CI specification contains only the base message definitions used for Property Exchange. Critical information and further rules for implementing Property Exchange are defined in the ***[MA05]*** M2-102-U Common Rules for Property Exchange specification.

Detailed information about the Header Data and Property for individual PE Resources, including all defined properties and semantics, is defined in specifications for each PE Resource.

All Devices which implement a PE Resource shall comply with the rules of this Section 8 Property Exchange, the rules of ***[MA05]*** M2-102-U Common Rules for MIDI-CI Property Exchange specification, and the rules of the specification for the PE Resource.

8.1 Property Exchange Inquiry and Negotiation Mechanism

Properties are exchanged by the following Common Property Exchange messages:

- Inquiry: Property Exchange Capabilities
- Reply to Property Exchange Capabilities
- 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 Exchange Addressing

The Device ID field of all Property Exchange messages shall be set to 0x0F (whole Function Block). Property Exchange mechanisms intended for individual Channels within a Function Block may be addressed by channelized properties within the Property Data.

8.3 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.

There is always at least one Chunk, even if it does not contain any Property Data. If there is no Property Data, then the Number of Chunks and Number of This Chunk shall both be set to a value of 0x0001 (See Section ***8.3.2***).

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 (starting the count from a value of 0x0001).

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.

8.3.1 Header Data in First Chunk Only

Header Data shall always be in the first Chunk only. Chunk numbers 2 and higher shall set the Length of Following Header Data to 0x0000 and shall not contain any Header Data.

8.3.2 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 0x0001 and Number of This Chunk shall be set to 0x0001.

8.3.3 Chunking Examples

Table 29 Chunking Examples: 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

8.4 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.

The Initiator may freely choose the value of the Request ID in the initial request message, but the value shall be unique among all Request IDs from this Initiator to the specific Responder.

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.

8.5 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.

Table 30 Inquiry: Property Exchange Capabilities Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	Destination 7F = to/from whole Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
30	Universal System Exclusive Sub-ID#2: Inquiry: Property Data Exchange Capabilities
1 byte	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
The following fields (except F7 End) were added in MIDI-CI Message Version 2	
1 byte	Property Exchange Major Version
1 byte	Property Exchange Minor Version
F7	End Universal System Exclusive

Property Exchange Major Version and Property Exchange Minor Version

The Property Exchange Major Version and Property Exchange Minor Version fields relate to different versions of the Common Rules for MIDI-CI Property Exchange ***[MA05]***. Updated versions of Common Rules for MIDI-CI Property Exchange may extend Resources, allow for greater compatibility, or simplified usage. Future versions of Common Rules for MIDI-CI Property Exchange will clearly list these changes and highlight which version they should report and how to address working with previous versions.

Table 31 Property Exchange Versions

Common Rules for MIDI-CI Property Exchange Version	Property Exchange Major Version	Property Exchange Minor Version
1.0/1.1*	0x00	0x00

** Version 1.1 contains editorial improvements over Version 1.0 and is the preferred reference.*

8.6 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.

Table 32 Reply to Property Exchange Capabilities Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	Destination 7F = to/from whole Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
31	Universal System Exclusive Sub-ID#2: Reply to Property Data Exchange Capabilities
1 byte	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
The following fields (except F7 End) were added in MIDI-CI Message Version 2	
1 byte	Property Exchange Major Version
1 byte	Property Exchange Minor Version
F7	End Universal System Exclusive

8.7 Inquiry: Get Property Data

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

Table 33 Inquiry Get Property Data Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	Destination 7F = to/from whole Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
34	Universal System Exclusive Sub-ID#2: Inquiry: Get Property Data
1 byte	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
01 00	Number of Chunks in Message (LSB first)
01 00	Number of This Chunk (count starts from 0x0001) (LSB first)
00 00	Length of Following Property Data (in this Chunk) (LSB first)*
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.*

8.8 Reply to Get Property Data

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

Table 34 Reply to Get Property Data Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	Source 7F = to/from whole Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
35	Universal System Exclusive Sub-ID#2: Reply to Get Property Data
1 byte	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)
2 bytes	Length of Following Property Data (in this Chunk) (LSB first)
nn bytes	Property Data
F7	End Universal System Exclusive

8.9 Inquiry: Set Property Data

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

Table 35 Inquiry Set Property Data Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	Destination 7F = to/from whole Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
36	Universal System Exclusive Sub-ID#2: Inquiry: Set Property Data
1 byte	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)
2 bytes	Length of Following Property Data (in this Chunk) (LSB first)
nn bytes	Property Data
F7	End Universal System Exclusive

8.10 Reply to Set Property Data

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

Table 36 Reply to Set Property Data Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	Source 7F = to/from whole Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
37	Universal System Exclusive Sub-ID#2: Reply to Set Property Data
1 byte	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
01 00	Number of Chunks in Message (LSB first)
01 00	Number of This Chunk (count starts from 0x0001) (LSB first)
00 00	Length of Following Property Data (in this Chunk) (LSB first)*
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.

8.11 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.

Table 37 Subscription Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	Destination

	7F = to/from whole Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
38	Universal System Exclusive Sub-ID#2: Subscription
1 byte	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)
2 bytes	Length of Following Property Data (in this Chunk) (LSB first)
nn bytes	Property Data
F7	End Universal System Exclusive

8.12 Reply to Subscription

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

Table 38 Reply to Subscription Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	Source 7F = to/from whole Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
39	Universal System Exclusive Sub-ID#2: Reply to Subscription
1 byte	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)

2 bytes	Number of This Chunk (count starts from 0x0001) (LSB first)
2 bytes	Length of Following Property Data (in this Chunk) (LSB first)
nn bytes	Property Data
F7	End Universal System Exclusive

8.13 Notify

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

See the Common Rules for Property Exchange specification for details.

Note: MIDI-CI ACK and NAK messages are generally more informative reporting mechanisms and are preferably used wherever applicable instead of this Notify message. See Sections **5.10** *and* **5.11**.

Table 39 Notify Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	Source 7F = to/from whole Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
3F	Universal System Exclusive Sub-ID#2: Notify
1 byte	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)
2 bytes	Length of Following Property Data (in this Chunk) (LSB first)
nn bytes	Property Data
F7	End Universal System Exclusive

9 Process Inquiry

Version 1.2 of MIDI-CI introduces a new category of MIDI-CI, Process Inquiry, which allows an Initiator to discover the current state of supported MIDI Messages in a Responder including:

- System Messages
- Channel Controller Messages
- Note Data Messages

9.1 Process Inquiry and Negotiation Mechanism

Properties are exchanged by the following Common Process Inquiry messages:

- Inquiry: Process Inquiry Capabilities
- Reply to Process Inquiry Capabilities
- Inquiry: MIDI Message Report
- Reply to MIDI Message Report
- End of MIDI Message Report

9.2 Inquiry: Process Inquiry Capabilities

The Process Inquiry (0x40) message allows the Initiator to detect the supported features from the Process Inquiry category. Process Inquiry Capabilities shall always set the Destination to 0x7F to address the Function Block (see Sections, ***3.1***, ***3.2.1***, and ***3.2.2***) (per-channel requests are not allowed).

Table 40 Inquiry: Process Inquiry Capabilities Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	Destination 7F = to/from whole Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
40	Universal System Exclusive Sub-ID#2: Inquiry: Process Inquiry Capabilities
1 byte	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
F7	End Universal System Exclusive

9.3 Reply to Process Inquiry Capabilities

A Responder which supports Process Inquiry and which receives an Inquiry: Process Inquiry Capabilities message shall send Reply to Process Inquiry message to the Initiator.

Table 41 Reply to Process Inquiry Capabilities Message Format

Value	Parameter
F0	System Exclusive Start

7E	Universal System Exclusive
7F	Destination 7F = to/from whole Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
41	Universal System Exclusive Sub-ID#2: Reply to Process Inquiry Capabilities
1 byte	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
1 byte	Process Inquiry Supported Features (bitmap)
F7	End Universal System Exclusive

Process Inquiry Supported Features

The Process Inquiry Supported Features bitmap is defined as follows:

Table 42 Inquiry: Process Inquiry Supported Features Bitmap Format

Bit	Description
D0	MIDI Message Report
D1..6	Reserved – No Messages Defined Yet

9.4 MIDI Message Report

The MIDI Message Report feature allows Initiators to request information from a Responder about the current state or values of those Responder's parameters which are currently controllable by MIDI Messages. This mechanism cannot be used to target any messages which are not stateful.

Use cases are (but not limited to):

- Query the current values of parameters which are settable by MIDI Controller messages.
- Query to find out which Program is currently active.
- Query to find out which Notes are currently actively playing.
- Query to find out the current song position of a sequence.
- Evaluate behavior of a device during MIDI compliance tests.

The MIDI Message Report Transaction proceeds with 4 stages:

1. An Initiator requests a MIDI Message Report using an Inquiry: MIDI Message Report request. Certain fields in the message declare target parameters based on the type of MIDI message which control those targets.
2. The Responder's reply consists of multiple messages, starting with the Reply to MIDI Message Report (Begin) message. This message contains bitmaps indicating which of the requested messages the device supports and is about to return in reply to the MIDI Message Report request.
3. The Responder reports the current state or value of all requested and supported MIDI parameters by sending the associated MIDI Messages with the current parameter values.
4. The Responder declares the end of the MIDI Message Report by sending an End of MIDI Message Report message.

Note that stage 3 above of this mechanism uses MIDI Messages in an atypical manner. In other common applications, MIDI Messages are usually sent as commands to set parameters in another receiving Device. But in this application, MIDI Messages sent in reply by a Responder are not intended to be commands to the Initiator, but instead are a report of the current values in the Responder. The Initiator may also and optionally use those messages as commands if the Initiator needs them for a specific application (such as synchronizing the value(s) between Initiator and Responder).

9.5 Inquiry: MIDI Message Report

An Initiator sends an Inquiry: MIDI Message Report message to request a MIDI data report from a Responder.

Bitmaps in the message allow the Initiator to specify which message states are being requested:

- System Messages
- Channel Controller Messages
- Note Data Messages

A Device may choose not to reply to all message requests in a MIDI Message Report, even if the Device implements the requested message. As an example, a sound generator which supports MIDI notes and per-note controllers is not required to report its internal note and per-note controller states.

Table 43 Inquiry: MIDI Message Report Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Device ID: Source or Destination (depending on type of message): 00–0F: To/from MIDI Channels 1-16 10-7D: Reserved 7E: To/from Group 7F: To/from Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
42	Universal System Exclusive Sub-ID#2: Inquiry: MIDI Message Report
1 byte	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
1 byte	Message Data Control
1 byte	Bitmap of requested System Messages D0: MTC Quarter Frame D1: Song Position D2: Song Select D3..7: reserved, set to 0
1 byte	Reserved for other System Messages, set to 0x00
1 byte	Bitmap of requested Channel Controller Messages D0: Pitchbend D1: Control Change D2: RPN / Registered Controller

	D3: NRPN / Assignable Controller D4: Program Change D5: Channel Pressure D6..7: reserved, set to 0
1 byte	Bitmap of requested Note Data Messages D0: Notes D1: Poly Pressure D2: Per-Note Pitchbend D3: Registered Per-Note Controller D4: Assignable Per-Note Controller D5..7: reserved, set to 0
F7	End Universal System Exclusive

An inquiry requesting a set of active Notes should only be used for compliance purposes.

In case of Channel Controller and Note Data Message types, an Initiator may specify a MIDI Channel in the Device ID field. An inquiry requesting System Messages shall have the Device ID set to 0x7F (Function Block). If Device ID is 0x7F when Channel Controller and/or Note Data Message types are requested, the Responder shall report all supported Groups and Channels.

The Responder shall return a MIDI-CI NAK Message if Device ID is set between 00-0F (channel 1-16) and that MIDI Channel is not in use on the Function Block (or if no Function Blocks exist, then if that MIDI Channel is not in use on the Device).

9.5.1 Message Data Control

An Initiator sets the Message Data Control field to configure the amount of data the Responder will return in response to a MIDI Message Report request.

Table 44 Message Data Control Values

Value	Description
0x00	Do not report any data, send Begin / End replies only. This enables the Initiator to query which types of MIDI messages the Responder is capable of reporting by a MIDI Message Report
0x01	Report only controllers with non-default values, active notes, and per-note controllers with non-default values.
0x02 - 0x7E	Reserved
0x7F	Full report of all requested and supported controllers, notes, and supported per-note controllers

Setting the Message Data Control value to 0x00 is useful for Initiators to detect Responders MIDI Message Report capabilities. This can especially be useful in automated MIDI compliance tests.

Setting the Message Data Control value to 0x01 is intended to reduce the amount of data being sent by the Responder. Many Controllers, especially Per-Note-Controllers, are typically set to their default values (e.g., 100 for Volume, 64 for Pan and Balance, 0 for most other controllers), such controllers can be omitted from the report. For a list of standardized default values see M2-113-UM MIDI-CI Profile: Default Control Change Mapping specification ***[MA07]***.

Setting the Message Data Control value to 0x7F requests a full report of all supported and requested messages. If a full report of note data is requested, a Responder should report Note On / Off messages for all active and inactive notes.

9.6 Reply to MIDI Message Report

If a Responder which supports Process Inquiry receives an Inquiry: MIDI Message Report, then Responder shall start its reply by sending a Reply to MIDI Message Report message.

The Reply to MIDI Message Report message has fields which contain the same MIDI Message bitmaps as the Inquiry: MIDI Message Report message. The Responder shall set values in the bitmaps to declare which of the types of messages it will be sending in its response.

For example: An Initiator may request Channel Pressure. However, the Responder cannot provide this so will set this bit to 0 to indicate that it will not be sending Channel Pressure data.

The Responder shall not set any additional bits and shall not report messages that were not requested.

The Responder shall set the Device ID field to the same value as in the Inquiry: MIDI Message Report message it received. If Device ID is 0x7E (Group) or 0x7F (Function Block), the Responder shall not send individual Reply to MIDI Message Report (Begin) and End Messages for each individual supported channel.

Table 45 Inquiry: Reply to MIDI Message Report Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Device ID: Source or Destination (depending on type of message): 00–0F: To/from MIDI Channels 1-16 10-7D: Reserved 7E: To/from Group 7F: To/from Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
43	Universal System Exclusive Sub-ID#2: Reply to MIDI Message Report
1 byte	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)
1 byte	Bitmap of requested System Messages D0: MTC Quarter Frame D1: Song Position D2: Song Select D3..7: reserved, set to 0
1 byte	Reserved for other System Messages, set to 0x00
1 byte	Bitmap of requested Channel Controller Messages D0: Pitchbend D1: Control Change D2: RPN / Registered Controller D3: NRPN / Assignable Controller D4: Program Change D5: Channel Pressure D6..7: reserved, set to 0
1 byte	Bitmap of requested Note Data Messages

	D0: Notes D1: Poly Pressure D2: Per-Note Pitchbend D3: Registered Per-Note Controller D4: Assignable Per-Note Controller D5..7: reserved, set to 0
F7	End Universal System Exclusive

9.7 MIDI Message Reporting

After the Responder declares the supported MIDI message(s) in Reply to MIDI Message Report (see Section ***9.6***) the Responder shall send the MIDI message(s) it has declared with the current value for each of the supported message(s).

9.7.1 Order of Message Reporting

Messages should be sent according to the following:

1. Messages should be sent in the order of messages declared in the Bitmap of requested System Messages D0-D7, Bitmap of requested Channel Controller Messages D0-D7, and Bitmap of requested Note Data Messages D0-D7 fields of the Inquiry: Reply to MIDI Message Report.
   For example, if Notes, Poly Pressure, and Per-Note Controllers are requested, then the state of Note On/Off for all notes will be reported first, followed by the state of Poly Pressure for all notes, and then followed by the state of Per-Note Controllers for all notes.
2. If the Device ID is set to 0x7E, all the channelized messages on a Channel should be sent before sending messages on the next Channel. Channelized messages should be sent in ascending Channel number order in the Group from Channel 1 through 16.
3. If the Device ID is set to 0x7F, messages should be sent for each Group according to the order defined in 1. and 2. above before sending messages on the next Group. Messages should be sent in ascending Group number for the Groups in the Function Block.

9.7.2 Reporting Rules for Specific Messages

The following messages have unique rules for reporting that type.

MTC Quarter Frame

The Responder shall send the complete set of eight quarter frame messages required to fully declare the current time.

Control Change

For each Control Change message recognized:

1. Control Change Index 0-63:
   A. If implemented as MSB+LSB then the Responder shall send Control Change 0-31 each followed by the matching LSB 32-63(0+32, 1+33, … etc.) with the current value of the Control Change.
   B. If 32-63 are not used as LSB for 0-32, then the Responder shall send Control Change 0-63 in index order with the current value of the Control Change.
2. Control Change 64-119,122: The Responder shall send the Control Change with the current value of the Control Change.
3. The Responder shall send only the Mode Messages for the current Mode (124, 125, 126 or 127) enabled. The Responder shall indicate the correct number of channels in Mono Mode On.

4. If in MIDI 1.0 Protocol: The Responder shall only send Control Change 0, 6. 32. 38 and 98–101 if these are (incorrectly) used as individual Controller Messages. These shall not be sent as part of Control Change reporting if they are in use as Bank Select, RPN and NRPN messages.

Registered Controller / RPN

MIDI 1 Protocol only: After all RPN's have been sent, the Responder shall send the Null Function Value RPN

Assignable Controller / NRPN

MIDI 1 Protocol only: After all NRPN's have been sent, the Responder shall send the Null Function Value NRPN

Program Change

In MIDI 1.0 Protocol:

If Bank Select is supported, the Responder shall send Bank Select MSB and LSB followed by a Program Change message.

If Bank Select is not supported, the Responder shall send only a Program Change.

In MIDI 2.0 Protocol the Responder shall send a Program Change message with Bank Select fields set properly.

Note On and Note Off

1a. In MIDI 1.0 Protocol: The Responder shall send a Note On message for each currently active note with the original velocity value.

1b. In MIDI 2.0 Protocol: The Responder shall send Note On messages for each currently active note with the original values of Attribute Type, Velocity, and Attribute Data field.

2. For every Note On message which has been sent, the Responder shall send a matching Note Off message prior to sending the End of MIDI Message Report.

9.8 End of MIDI Message Report

After all requested messages have been reported, the Responder shall complete the reply by sending an End of MIDI Message Report message.

Table 46 End of MIDI Message Report Message Format

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Device ID: Source or Destination (depending on type of message): 00–0F: To/from MIDI Channels 1-16 7E: To/from Group 7F: To/from Function Block
0D	Universal System Exclusive Sub-ID#1: MIDI-CI
44	Universal System Exclusive Sub-ID#2: End of MIDI Message Report
1 byte	MIDI-CI Message Version/Format
4 bytes	Source MUID (LSB first)
4 bytes	Destination MUID (LSB first)

F7	End Universal System Exclusive

9.8.1 Incomplete MIDI Message Report

If a Responder needs to abort a MIDI Message Report or has an error, the Responder shall send a MIDI-CI NAK with Status Code 0x20 (Terminate Inquiry) or 0x40 (Error occurred, please retry) See Section ***5.11***. If an Initiator receives a NAK, the Initiator shall assume that the MIDI Message Report Transaction has ended, even if an End of MIDI Message Report has not been received.

9.8.2 Potential Issues with Ordering

An incorrect MIDI implementation might change the ordering of System Exclusive messages and Channel Voice Messages in a MIDI Message stream. This could create problems when bracketing MIDI Message Reports (comprised of Channel Voice Messages) with the messages described here (System Exclusive Messages). Devices shall process all MIDI messages in the order received to avoid such problems.

Appendix A: 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 Sections ***5.9.1*** and ***5.9.2***). 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.

A.1 Device Design

The chance 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 a 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 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.

A.1.1 Using Product Instance Id

If a Device receives a MIDI-CI Discovery message with a MUID that matches its own MUID and Device identifiers, it can use the Product Instance ID to differentiate between 2 scenarios:

1. An echo back of its own message (matching Product Instance Id
2. A MUID collision with another Device of the same model (different Product Instance Id)

A.1.2 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 B: Version Considerations for MIDI-CI Specification Updates

B.1 Increasing the Message Format Version

The MIDI-CI Message Minor Format Version number shall be increased whenever a MIDI-CI specification update does one or more of the following:

- add new messages in existing categories unless the existence of the new messages can be discovered, and the new message does not change the function of other messages in the category.
- extend existing messages.
- deprecate messages.

The MIDI-CI Message Minor Format Version number should be increased whenever a MIDI-CI specification update does one or more of the following if AMEI/MA feels the impact is sufficiently significant:

- define previously reserved fields, bits in a bitmap, or values.

The MIDI-CI Message Minor Format Version number may be increased whenever a MIDI-CI specification update does one or more of the following if the working group feels the impact is sufficiently significant:

- introduce a new MIDI-CI message category.

The MIDI-CI Message Format Version number may stay the same if an update only introduces editorial changes.

A new MIDI-CI Message Major Version shall be introduced whenever a MIDI-CI specification update does one or more of the following which are likely to be incompatible with prior versions:

- change mechanisms of previously defined messages.
- change definitions of previously defined message fields.
- change data format of previously defined message fields.
- insert new fields between previously defined message fields.
- deprecate or remove messages.

The goal is that for every given Message Format Version, the fields and the total length of every message are well defined. This facilitates compliance testing, and it will prevent implementations which add accidental or private data to MIDI-CI messages.

Increasing the MIDI-CI message version applies to all messages, regardless of whether they were changed or not.

In general, the MIDI-CI specification document version is independent from the Message Format Version.

B.2 Adding New Messages or Categories

The definition of every new message shall mention the initial MIDI-CI Message Format Version when this message was introduced. This also applies to all messages that are added with a new MIDI-CI category. The specification text of an added category shall also specify the initial message version.

B.3 Modifying Existing Messages

Modifying existing messages should only add fields. Existing fields should not be modified.

When a specification update extends an existing message, the specification shall clearly specify which fields were added in this MIDI-CI message version as has been done for some messages in this version of specification.

B.4 Removing and Deprecating Messages

When a message is removed or deprecated, the specification shall either include the definition of the message, as defined in the last revision before the message was removed or deprecated, or declare which prior version of the

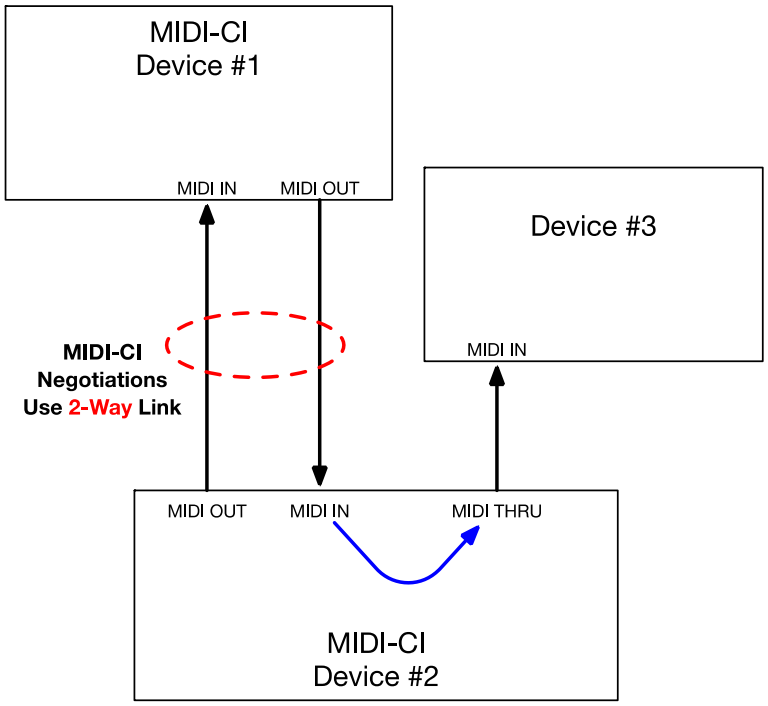
specification contains the full message definition. The message version of when it was removed or deprecated shall be clearly specified.

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 errors. A MIDI merger Device might also introduce 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.

Figure 5 MIDI Thru Limitation

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:

0x00-1F Reserved

0x20-2F Profile Configuration Messages

0x30-3F Property Exchange Messages

0x40-4F Process Inquiry

0x50-6F Reserved

0x70-7F Management Messages

Sub-ID#2	Message Type	Message Source	Addresses
Category 0: Reserved Space			
0x00-0x0F	Reserved		
Category 1: Protocol Negotiation (all messages deprecated as of version 1.2 of MIDI-CI)			
0x10	Initiate Protocol Negotiation		
0x11	Reply to Initiate Protocol Negotiation		
0x12	Set New Selected Protocol		
0x13	Test New Protocol Initiator to Responder		
0x14	Test New Protocol Responder to Initiator		
0x15	Confirmation Protocol Established		
0x16-1F	Reserved		
Category 2: Profile Configuration Messages			
0x20	Profile Inquiry	Initiator	0x00-0F, 0x7E, 0x7F
0x21	Reply to Profile Inquiry	Responder	0x00-0F, 0x7E, 0x7F
0x22	Set Profile On	Initiator	0x00-0F, 0x7E, 0x7F
0x23	Set Profile Off	Initiator	0x00-0F, 0x7E, 0x7F
0x24	Profile Enabled Report	Initiator or Responder	0x00-0F, 0x7E, 0x7F
0x25	Profile Disabled Report	Initiator or Responder	0x00-0F, 0x7E, 0x7F
0x26	Profile Added Report	Responder	0x00-0F, 0x7E, 0x7F
0x27	Profile Removed Report	Responder	0x00-0F, 0x7E, 0x7F
0x28	Profile Details Inquiry	Initiator	0x00-0F, 0x7E, 0x7F
0x29	Reply to Profile Details Inquiry	Responder	0x00-0F, 0x7E, 0x7F
0x2A-2E	Reserved		
0x2F	Profile Specific Data	Initiator or Responder	0x00-0F, 0x7E, 0x7F
Category 3: Property Exchange Messages			
0x30	Inquiry: Property Exchange Capabilities	Initiator	0x7F
0x31	Reply to Property Exchange Capabilities	Responder	0x7F
0x32	Inquiry: Has Property Data (Reserved)		0x7F
0x33	Reply to Has Property Data (Reserved)		0x7F
0x34	Inquiry: Get Property Data	Initiator	0x7F
0x35	Reply to Get Property Data	Responder	0x7F
0x36	Inquiry: Set Property Data	Initiator	0x7F
0x37	Reply to Set Property Data	Responder	0x7F

0x38	Subscription	Initiator or Responder	0x7F
0x39	Reply to Subscription	Initiator or Responder	0x7F
0x3A-3E	Reserved		0x7F
0x3F	Notify	Initiator or Responder	0x7F
Category 4: Process Inquiry			
0x40	Inquiry: Process Inquiry Capabilities	Initiator	0x7F
0x41	Reply to Process Inquiry Capabilities	Responder	0x7F
0x42	Inquiry: MIDI Message Report	Initiator	0x00-0F, 0x7E, 0x7F
0x43	Reply to MIDI Message Report	Responder	0x00-0F, 0x7E, 0x7F
0x44	End of MIDI Message Report	Responder	0x00-0F, 0x7E, 0x7F
0x46-4F	Reserved		
Categories 5-6: Reserved Space			
0x50-6F	Reserved		
Category 7: Management Messages			
0x70	Discovery	Initiator	0x7F
0x71	Reply to Discovery	Responder	0x7F
0x72	Inquiry: Endpoint Information	Initiator	0x7F
0x73	Reply to Endpoint Information	Responder	0x7F
0x74-7C	Reserved		
0x7D	MIDI-CI ACK	Initiator or Responder	0x00-0F, 0x7E, 0x7F
0x7E	Invalidate MUID	Initiator or Responder	0x7F
0x7F	MIDI-CI NAK	Initiator or Responder	0x00-0F, 0x7E, 0x7F

Appendix E: 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 these general requirements:
    - Support System Exclusive message lengths of at least 128 bytes. Support System Exclusive message lengths of at least 512 bytes if either Profile Configuration or Property Exchange is supported.
    - Respond to MIDI-CI Discovery messages
    - Respond to Invalidate MUID Messages
    - Be able to send a NAK message when appropriate

2. MIDI-CI Devices shall implement multiple features, as defined in the MIDI-CI specification, 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 (Profile Configuration, Property Exchange, or Process Inquiry) are encouraged to use MIDI-CI. The central feature of the minimum requirements above is the ability to 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.

http://www.amei.or.jp

https://www.midi.org