MIDI 2.0 Specification Overview

with Minimum Requirements

MIDI Association Document: M2-100-U

Document Version 1.1
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 Association Document M2-100-U
MIDI 2.0 Specification Overview

This document defines the specific collection of MA/AMEI specifications that collectively comprise the MIDI 2.0 Specification.

The document also defines minimum requirements for Devices to claim MIDI 2.0 compatibility.

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
Feb. 20, 2020	1.0	Initial release
June 15, 2023	1.1	Minor Update recognizing changes in other core MIDI 2.0 specifications.

Contents

1 Introduction

1.1 Executive Overview

MIDI 2.0 is an extension of MIDI 1.0. It does not replace MIDI 1.0 but builds on the core principles, architecture, and semantics of MIDI 1.0.

MIDI has grown and continued to serve users well since its inception in 1983 as the worldwide standard method of musical communication and control. The MIDI industry, through the standardization work of the Association of Music Electronics Industry (AMEI) and the MIDI Association (MA), has greatly expanded the capabilities and market reach of MIDI in the years and decades since 1983.

MIDI 1.0 has some limitations for future expansion which are addressed by MIDI 2.0. A key concept in MIDI 2.0 is the reliance on bidirectional communication for devices to better auto-configure for more tight interoperability. A new MIDI message data format allows more expressive musical control and room to add thousands of new MIDI messages in the future.

This specification informs the reader about the core specifications of MIDI 2.0 and defines the minimum requirement for a device to claim that implements MIDI 2.0.

1.2 MIDI 2.0 Overview

This document defines the specific collection of MA/AMEI specifications that collectively comprise the core mechanisms of MIDI 2.0. MIDI 2.0 is not a stand-alone specification. Manufacturers and developers must also have a thorough understanding of MIDI 1.0 to implement MIDI 2.0.

The document also defines minimum requirements for Devices to claim MIDI 2.0 compatibility.

1.2.1 Core Specifications of MIDI 2.0

Along with this MIDI Specifications Overview document, the following foundational MIDI Specifications define the core features of MIDI 2.0:

1. M2-101-UM MIDI Capability Inquiry (MIDI-CI), Version 1.2 ***[MA02]***
2. M2-102-U Common Rules for MIDI-CI Profiles, Version 1.1 ***[MA03]***
3. M2-103-UM Common Rules for Property Exchange, Version 1.1 ***[MA04]***
4. M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol, Version 1.1 ***[MA05]***

Each of these specifications is described in Section ***2***

1.2.2 Minimum Requirements of MIDI 2.0

There are two distinct paths for Devices to achieve the minimum requirements to MIDI 2.0 compliance. Devices must implement MIDI Capability Inquiry (MIDI-CI) ***[MA02]*** plus some other defined features and/or implement the Universal MIDI Packet (UMP) ***[MA05]*** plus some other defined features.

See Section ***5*** for the full definition of the requirements.

1.3 References

1.3.1 Normative References

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

[MA03] ***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/

[MA04] ***M2-103-UM Common Rules for Property Exchange***, Version 1.1, Association of Musical Electronics Industry, http://www.amei.or.jp/, and The MIDI Association, https://www.midi.org/

[MA05] ***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/

1.3.2 Informative References

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

1.4 Terminology

1.4.1 Definitions

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

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

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 ***[MA05]***.

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.

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 ***[MA05]***.

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 ***[MA05]***.

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

MIDI-CI: MIDI Capability Inquiry ***[MA02]***, a 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 Endpoint: An entity 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 the 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.

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. See the MIDI-CI specification ***[MA02].***

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 ***[MA02]*** and the Common Rules for MIDI-CI Profiles ***[MA03]***.

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 Properties from another device. See the MIDI-CI specification ***[MA02]*** and the Common Rules for Property Exchange ***[MA04]***.

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 ***[MA05]***.

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.

UMP: Universal MIDI Packet, see ***[MA05]***.

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

UMP Format: Data format for fields and messages in the Universal MIDI Packet, see ***[MA05]***.

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 ***[MA05]***.

1.4.2 Reserved Words and Specification Conformance

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

Table 2 Words Relating to Specification Conformance

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

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

Table 3 Words Not Relating to Specification Conformance

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

2 Core Components of MIDI 2.0

2.1 Discovery, Bidirectional Autoconfiguration, and Expanded Data Format

MIDI 2.0 builds on the core principles, architecture, and semantics of MIDI 1.0. Primary features are auto-configuration, enabled by bidirectional connections enabling devices to discover details about other connected devices, and an expanded data format for higher resolution with extensibility to define many new messages in the future.

- MIDI Capability Inquiry allows devices to discover each other and learn which MIDI 2.0 auto-configuration mechanisms each one supports, enabling increased interoperability.
- The Universal MIDI Packet defines an expanded data format. New messages allow devices to discover each other's topology and data support features for communicating with high-resolution messages.

These fundamental definitions for these features are in specifications listed in Section ***2.2***.

2.2 Four Core Documents

MIDI 2.0 includes four main documents that expand on the capabilities of MIDI 1.0. These four documents collectively, along with this MIDI 2.0 Specification document, define the core architecture of MIDI 2.0 and its connection to MIDI 1.0. None of these documents are stand-alone but are extensions of MIDI 1.0. Manufacturers and developers must have a thorough understanding of MIDI 1.0 to implement MIDI 2.0.

MIDI 2.0 features are defined in the following documents:

1. MIDI Capability Inquiry (MIDI-CI) ***[MA02]***
2. Common Rules for MIDI-CI Profiles ***[MA03]***
3. Common Rules for MIDI-CI Property Exchange ***[MA04]***
4. Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol ***[MA05]***

2.3 MIDI Capability Inquiry (MIDI-CI)

MIDI-CI defines a discovery mechanism and an architecture that allows Devices with bidirectional communication to agree to use the extended capabilities of MIDI 2.0 for autoconfiguration and interoperability.

Goals of MIDI-CI design:

- Allow a Sender to know the capabilities of a Receiver.
- Enable easier configuration between Devices using auto-configuration Profiles.
- Define method for Getting and Setting a wide range of Device properties.

MIDI-CI delivers on these goals with 3 main areas of functionality:

- **Profile Configuration**: use auto-configuration Profiles.
- **Property Exchange**: Get & Set Device Property Data using JSON.
- **Process Inquiry**: Discover the current state/value of supported MIDI Messages

MIDI-CI defines the core messages for Profile Configuration and Property Exchange. Further details of and rules for implementing Profile Configuration and Property Exchange are found in "Common Rules" documents. See Sections ***2.4*** and ***2.5***.

The MIDI-CI specification includes the complete specification for Process Inquiry and there is no separate Common Rules document for Process Inquiry.

MIDI-CI Discovery mechanisms must be enacted before any other features of MIDI-CI can operate.

See the MIDI Capability Inquiry Specification (MIDI-C) *[MA02]* for more details.

2.4 Common Rules for MIDI-CI Profiles

Profiles are a beneficial component in enabling intelligent auto-configuration.

A MIDI-CI Profile is a defined set of MIDI messages and implementation rules to achieve a particular purpose or to suit a particular application. In addition to defining responses to MIDI messages, a Profile may optionally also define other Device functionality requirements. In general, Profiles define Receiver implementation. A Profile definition may also imply or require MIDI implementation of a Sender.

The most successful legacy MIDI feature that is similar to a Profile in the first three decades of MIDI has been General MIDI. GM allows Devices to "know" that a defined set of sounds is available at particular Program Change locations, that the Device receives on all 16 MIDI channels, that a Drum set is on channel 10, and that there is a defined response to a chosen set of MIDI messages.

This kind of knowledge shared between Devices allows those Devices to configure a more integrated level of control with increased predictability of the results that will come from sending related MIDI messages. MIDI-CI Profiles are also intended to allow more integrated cooperation between Devices.

While GM is the best model of a successful profile concept prior to MIDI-CI, it does not take MIDI-CI or a two-way communication into account. There is a "GM On" message but no reply from the Receiver. New MIDI-CI Profiles take advantage of two-way communication. This document defines how specific Profile specifications should be written and how Devices that are compatible with MIDI-CI Profile Configuration should use Profiles.

See the Common Rules for MIDI-CI Profiles *[MA03]* for more details.

2.5 Common Rules for MIDI-CI Property Exchange

Property Exchange is a set of mechanisms to get and set Device property data using JSON transmitted via MIDI-CI Universal System Exclusive messages.

Property Exchange allows Devices to auto map controllers, choose programs, change state and also provide visual editors to DAW's without any prior knowledge of the Device or specially crafted software. This means that Devices could work on a wide range of systems such as desktop operating systems, mobile devices, and web browsers and may provide tighter integrations with DAWs and hardware controllers.

Property Exchange provides a common way for Devices to work together by providing defined schema for describing how Property Data is transferred.

Specially written software might provide a more unique experience by relying on custom software and SysEx. However, by using Property Exchange, the longevity and accessibility of equipment is no longer restricted by underlying system upgrades or new platforms that appear in the marketplace.

See the Common Rules for MIDI-CI Property Exchange *[MA04]* for more details.

2.6 Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol

The Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification defines a new data format for MIDI 1.0 Protocol messages, MIDI 2.0 Protocol messages, and a foundation for the expansion of MIDI.

The UMP Format includes discovery mechanisms for UMP Endpoints which must be enacted before using other features of the Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification. UMP discovery enables increased interoperability, such as bidirectional transactions for switching between MIDI 1.0 Protocol and MIDI 2.0 Protocol.

The data format of the UMP adds 16 Groups, each containing an independent set of System Messages and 16 Channels that are equivalent to MIDI 1.0 Channels. The Universal MIDI Packet format adds a Jitter Reduction Timestamp mechanism. A Timestamp can be prepended to any MIDI message for improved timing accuracy.

The MIDI 2.0 Protocol provides extended data resolution for all Channel Voice Messages. MIDI 2.0 makes some messages easier to use, by aggregating combination messages into one atomic message. It adds new properties to

several Channel Voice Messages. New Channel Voice Messages are added to provide increased Per-Note control and musical expression.

New data messages include System Exclusive 8 and Mixed Data Set. The System Exclusive 8 message is very similar to MIDI 1.0 System Exclusive but with 8 bit data format. The Mixed Data Set Message is used to transfer large data sets, including non-MIDI data.

New messages are defined to express properties which were previously non-MIDI "Meta Events" in version 1 of Standard MIDI File formats. All properties in version 2 of Standard MIDI File formats can be exchanged with MIDI messages in the UMP Format.

The Universal MIDI Packet format includes a large, reserved space for future extensibility.

See the Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol Specification *[MA05]* for more details.

3 MIDI Transports

3.1 MIDI-CI on MIDI 1.0 and MIDI 2.0 Capable Transports

One of the core goals of MIDI 2.0 was to allow addition of MIDI 2.0 bi-directional communications to improve user experience even on MIDI 1.0 transports.

MIDI-CI Discovery, Profiles, Property Exchange, and Process Inquiry are implemented using System Exclusive messages. Therefore, MIDI-CI can be implemented on any existing MIDI 1.0 transport (5 PIN DIN, USB MIDI 1.0, RTP MIDI, BLE MIDI, Web MIDI, etc) and on MIDI 2.0 capable transports.

3.2 UMP Messages on MIDI 2.0 Capable Transports

All MIDI 2.0 capable transports express MIDI messages in the Universal MIDI Packet Format. All features defined in the Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification require MIDI 2.0 capable transports with the UMP data format.

4 Standard MIDI File Version 2 Format

Version 2 of Standard MIDI File formats express MIDI messages in the Universal MIDI Packet Format.

5 Minimum Compatibility Requirements of MIDI 2.0

Any Device which claims MIDI 2.0 compatibility shall implement either A or B, or both A and B:

A. MIDI-CI* to at least its minimum requirements, including discovery mechanisms, plus any one or more of the following features:

- One or more Profiles controllable by MIDI-CI Profile Configuration messages.
- Any Property Data exchange by MIDI-CI Property Exchange messages.
- Any Process Inquiry exchange by MIDI-CI Process Inquiry messages.

B. The UMP Data Format** to at least its minimum requirements, including discovery mechanisms, plus any one or more of the following features:

- MIDI 2.0 Channel Voice Messages as defined by the Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol.
- Jitter Reduction Timestamps as defined by the Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol.
- System Exclusive 8 as defined by the Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol.
- Mixed Data Set as defined by the Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol.

Note: MIDI-CI v1.2 requires the support of its discovery mechanisms. For a list of the complete minimum requirements to implement MIDI-CI see the M2-101-UM MIDI Capability Inquiry (MIDI-CI) Version 1.2 specification **[MA02].*

***Note: UMP and MIDI 2.0 Protocol v1.1 requires the support of its UMP Endpoint discovery mechanisms. For a list of the complete minimum requirements to implement UMP Format, see the M2-104-UM UMP and MIDI 2.0 Protocol Version 1.1 specification **[MA05]**.*

6 Notable Changes in Core MIDI Specifications

6.1 MIDI 2.0 Minimum Compatibility Requirements

The requirements for Devices to claim MIDI 2.0, defined in Section ***5***, have been updated. These updates are triggered by a set of changes in the MIDI Capability Specification (MIDI-C) ***[MA02]*** and the Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol Specification ***[MA05]*** as described in Section ***6.2***.

6.2 UMP Endpoints, Function Blocks and Protocols

The concepts of UMP Endpoints and Function Blocks have been added or further defined, with new Inquiry and Reply messages and mechanisms, in the UMP Format and MIDI 2.0 Protocol Specification ***[MA05]***. Those changes include mechanisms defined to allow Devices to select or change Protocols for subsequent communication. Therefore, the Protocol Negotiation mechanisms and messages originally defined in early versions of MIDI-CI ***[MA02]*** have been deprecated. MIDI-CI addressing has also been updated to define how MIDI-CI Transactions may be addressed between Function Blocks.

6.3 Backward and Forward Compatibility

The MIDI Capability Specification (MIDI-C) ***[MA02]*** and the Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol Specification ***[MA05]*** have added mechanisms for version control and to assist in maintaining compatibility across revisions of core MIDI 2.0 specifications.

6.4 Ongoing Expansion and Updates

MIDI 2.0 is a growing set of specifications. For further updates, changes, and additions to MIDI specifications, see the latest revisions of those individual specification documents. Additional specification documents, including Profiles, PE Resources, and others will be published by Association of Musical Electronics Industry, *http://www.amei.or.jp/*, and The MIDI Association, *https://www.midi.org/*.

7 Steps to Use MIDI 2.0

Following are steps to implement key features of MIDI 2.0. Devices which implement MIDI-CI on a MIDI 1.0 transport can skip Steps 2, 3, and 4 (messages used in steps 2, 3, and 4 require a UMP-capable transport and are never sent over a MIDI 1.0 transport).

Table 4 Steps to Use MIDI 2.0

Step	Mechanism	Action or Messages	Result or Information Learned
1	Connect Devices	Get Transport layer device descriptors	• Information for Transport Connection. • Device Name.
2	UMP Endpoint Discovery	• Endpoint Discovery • Endpoint Info Notification • Device Identity Notification • Endpoint Name Notification • Product Instance Id Notification	• Version of UMP supported. • Number of Function Blocks. • Device Info (Manufacturer, Model, etc.). • Endpoint Name. • Which Protocols are supported. • Whether JR timestamps are supported. • Instance Id (usually serial number) to unify multiple Function Blocks.
3	Select Protocol	• Stream Configuration Request • Stream Configuration Notification	• Request Endpoint Protocol and JR Timestamps on/off • Endpoint's Active Protocol is selected and JR Timestamps set to on/off
4	Function Blocks Discovery	• Function Block Discovery • Function Block Info Notification • Function Block Name Notification	• Which Function Blocks are currently active. • Which Groups are currently members of a Function Block. • If the Function Block is bidirectional. • If a Function Block supports MIDI-CI and version of MIDI-CI. • If a Function Block is a connection to a MIDI 1.0 device and bandwidth. • Name of a Function Block.
5	MIDI-CI Discovery	• MIDI-CI Discovery • Reply to Discovery	• Uniquely identify a device by MUID and Device Info (Manuf, Model, etc.). • Number (Id) of Function Block to unify with MIDI-CI device. • Max Rx SysEx size. • Categories of MIDI-CI Supported.
6	Use MIDI-CI	• Profile Configuration Messages • Property Exchange Messages • Process Inquiry Messages	• Supported Profiles • Supported PE Resources • Device parameter states
7	Use MIDI	• System Messages • Channel Voice Messages • Etc.	

http://www.amei.or.jp

https://www.midi.org