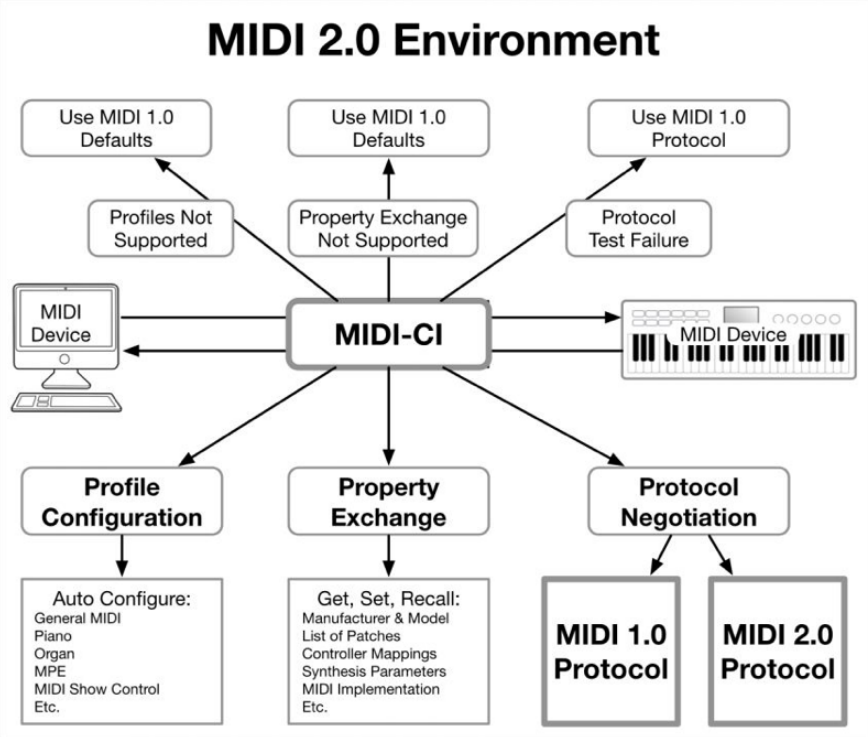
Details about MIDI 2.0™, MIDI-CI, Profiles and Property Exchange

Press Releases, MIDI History, MIDI News The MIDI Association



MIDI 2.0 specifications adopted at Winter NAMM 2020

On Sunday, January 19, 2020 at the Annual Meeting of the MIDI Manufacturers Association, the complete suite of MIDI 2.0 specifications were adopted unanimously by the MMA members in attendance .

These five core documents serve as the basis for the future expansion of MIDI.

The MIDI specifications adopted at Winter NAMM included:

- MIDI Capability Inquiry (update)
- Universal MIDI Packet & MIDI 2.0 protocol
- Common Rules for MIDI-CI Profiles
- Common Rules for MIDI-CI Property Exchange
- MIDI 2.0 Specifications Overview

The specifications are now being edited to their final format for official signing by AMEI (the Association of Music Electronics Industry, the Japanese MIDI standards organization) and then will be available for public download.

Introduction to MIDI 2.0™

Back in 1983, musical instrument companies that competed fiercely against one another nonetheless banded together to create a visionary specification—MIDI 1.0, the first universal Musical Instrument Digital Interface. Nearly four decades on, it's clear that MIDI was crafted so well that it has remained viable and relevant. Its ability to join computers, music, and the arts has become an essential part of live performance, recording, smartphones, and even stage lighting. Now, MIDI 2.0 takes the specification even further, while retaining backward compatibility with the MIDI 1.0 gear and software already in use. Here's why MIDI 2.0 is the biggest advance in music technology in decades.

MIDI 2.0 Means Two-way MIDI Conversations

MIDI 1.0 messages went in one direction: from a transmitter to a receiver. MIDI 2.0 is bi-directional and changes MIDI from a monologue to a dialog. For example, with the new MIDI-CI (Capability Inquiry) messages, MIDI 2.0 devices can talk to each other, and auto-configure themselves to work together. They can also exchange information on functionality, which is key to backward compatibility—MIDI 2.0 gear can find out if a device doesn't support MIDI 2.0, and then simply communicate using MIDI 1.0.

Higher Resolution, More Controllers and Better Timing

To deliver an unprecedented level of nuanced musical and artistic expressiveness, MIDI 2.0 re-imagines the role of performance controllers, the aspect of MIDI that translates human performance gestures to data computers can understand. Controllers are now easier to use, and there are more of them: over 32,000 controllers, including controls for individual notes. Enhanced, 32-bit resolution gives controls a smooth, continuous, "analog" feel. New Note-On options were added for articulation control and precise note pitch. In addition, dynamic response (velocity) has been upgraded. What's more, major timing improvements in MIDI 2.0 can apply to MIDI 1.0 devices—in fact, some MIDI 1.0 gear can even "retrofit" certain MIDI 2.0 features.

Profile Configuration

MIDI gear can now have Profiles that can dynamically configure a device for a particular use case. If a control surface queries a device with a "mixer" Profile, then the controls will map to faders, panpots, and other mixer parameters. But with a "drawbar organ" Profile, that same control surface can map its controls automatically to virtual drawbars and other keyboard parameters—or map to dimmers if the profile is a lighting controller. This saves setup time, improves workflow, and eliminates tedious manual programming.

Property Exchange

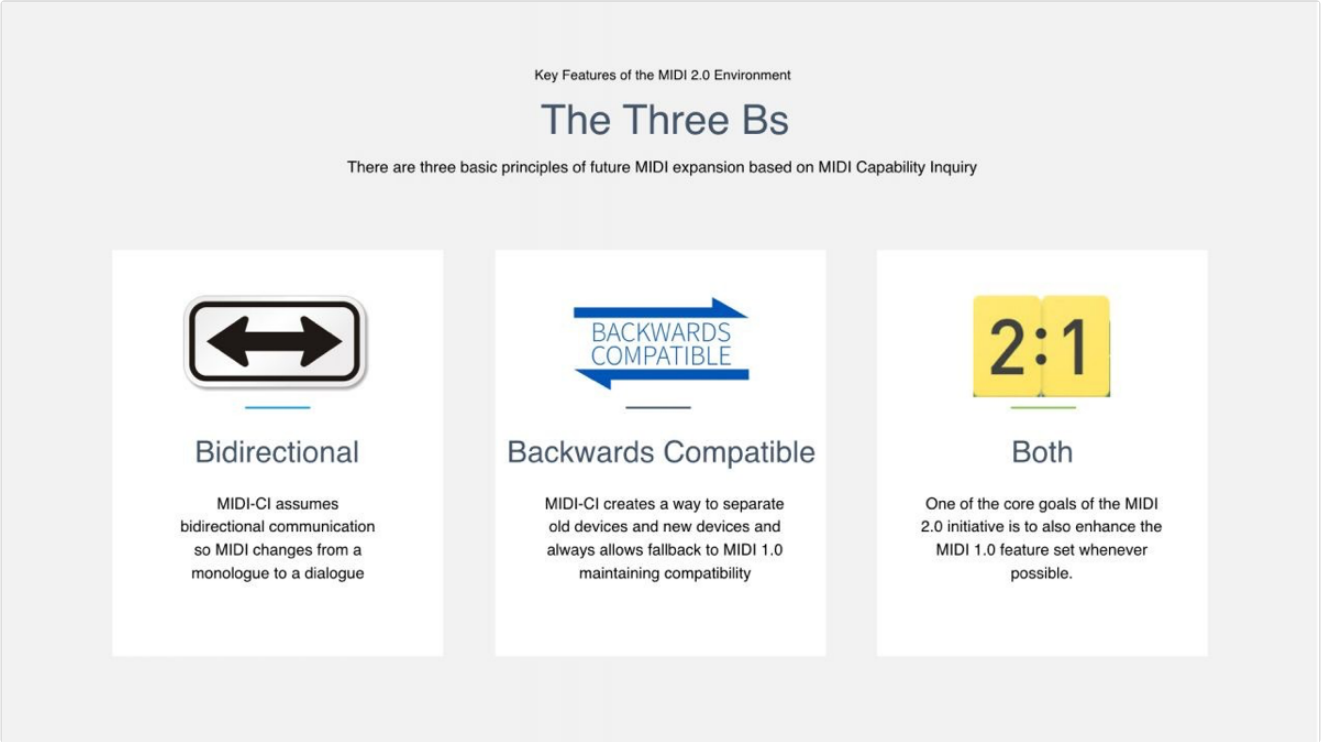
While Profiles set up an entire device, Property Exchange messages provide specific, detailed information sharing. These messages can discover, retrieve, and set many properties like preset names, individual parameter settings, and unique functionalities—basically, everything a MIDI 2.0 device needs to know about another MIDI 2.0 device. For example, your recording software could display everything you need to know about a synthesizer onscreen, effectively bringing hardware synths up to the same level of recallability as their software counterparts.

Built for the Future.

MIDI 2.0 is the result of a global, decade-long development effort. Unlike MIDI 1.0, which was initially tied to a specific hardware implementation, a new Universal MIDI Packet format makes it easy to implement MIDI 2.0 on any digital transport (like USB or Ethernet). To enable future applications that we can't envision today, there's ample space reserved for brand-new MIDI messages.

Further development of the MIDI specification, as well as safeguards to ensure future compatibility and growth, will continue to be managed by the MIDI Manufacturers Association working in close cooperation with the Association of Musical Electronics Industry (AMEI), the Japanese trade association that oversees the MIDI specification in Japan.

MIDI will continue to serve musicians, DJs, producers, educators, artists, and hobbyists—anyone who creates, performs, learns, and shares music and artistic works—in the decades to come.



1.1 MIDI Capability Inquiry (MIDI-CI)

The additional capabilities that MIDI 2.0 brings to devices are enabled by MIDI-CI. The basic idea is that if devices have a bidirectional connection, they can exchange their capabilities with each other. Devices can share their configuration and what MIDI functions are supported.

Devices use a bidirectional link to configure MIDI features when both devices agree to support that feature. MIDI-CI discovers and configures device features using 3 categories of inquiry: Profile Configuration, Property Exchange, and Protocol Negotiation.

If a device does not support any new features, it uses the MIDI 1.0 as usual. Devices connected to that device will continue to use MIDI 1.0 in communication with that device.

Expanding MIDI with new features requires a new protocol with extended MIDI messages. To protect backwards compatibility in an environment with expanded features, devices need to confirm the capabilities of other connected devices. When 2 devices are connected to each other, they use MIDI 1.0 and confirm each other's

capabilities before using expanded 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: Expanding MIDI while Protecting Backwards Compatibility:

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

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

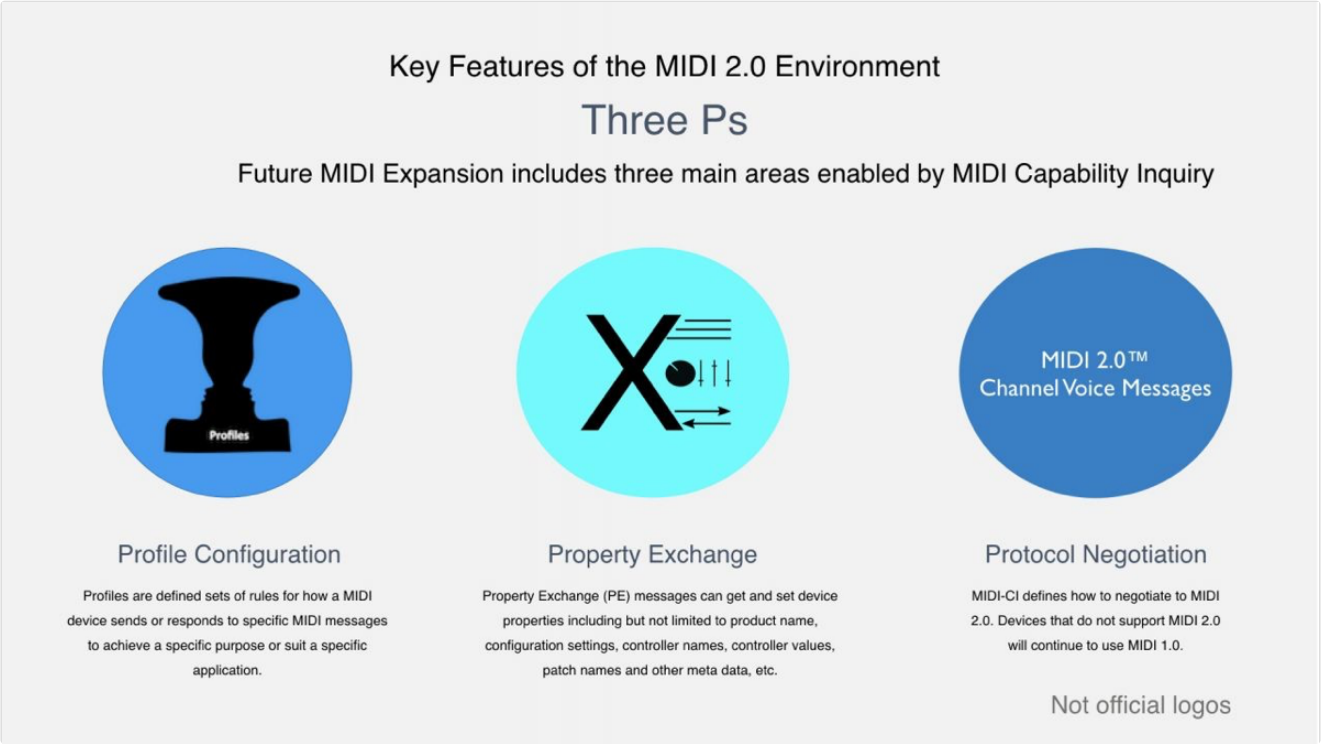
MIDI-CI assumes and requires bidirectional communication. Once a MIDI-CI connection is established between devices, query and response messages define what capabilities each device has.

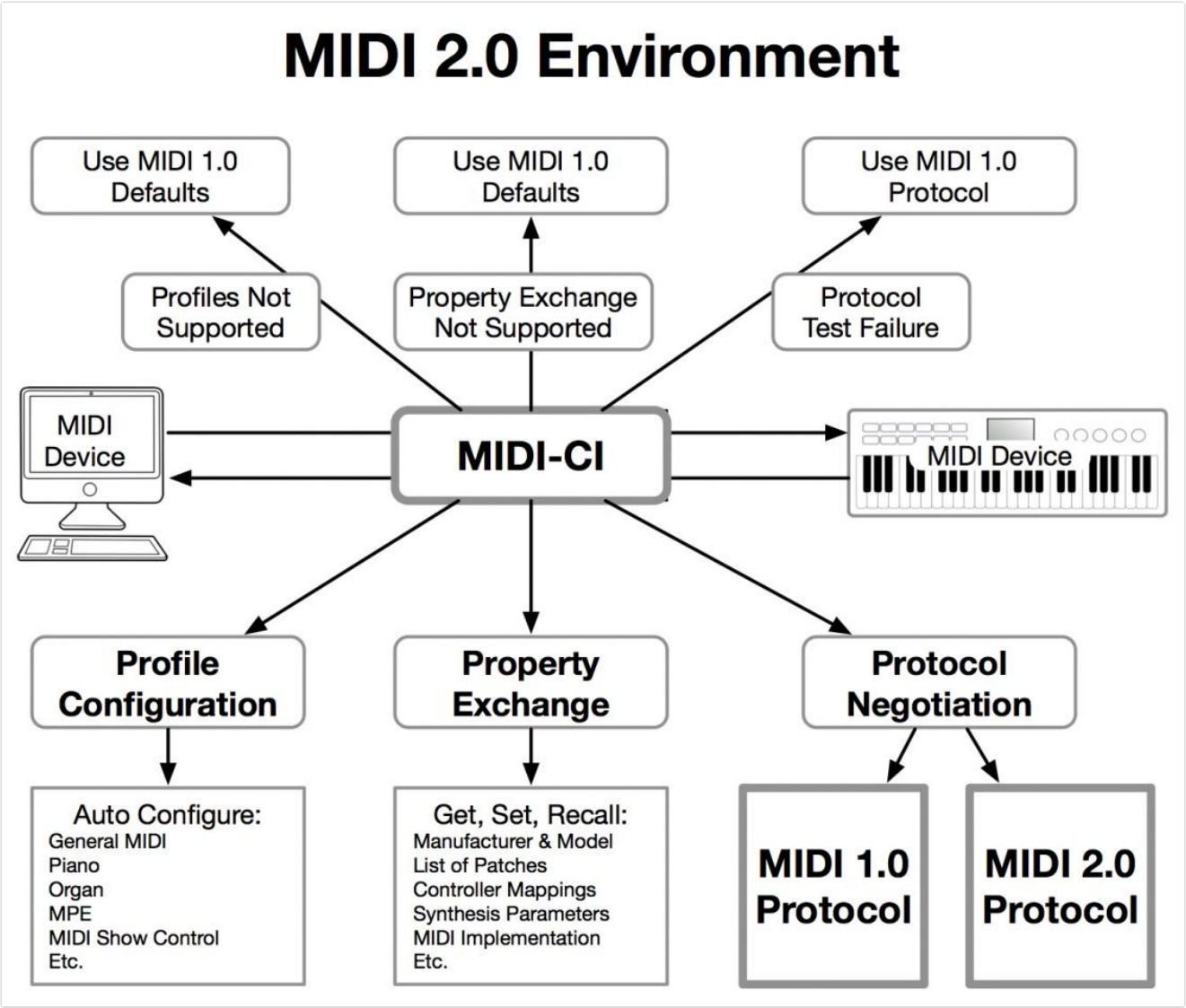
MIDI-CI then negotiates or auto-configures to use those features that are common between the devices. MIDI-CI provides test mechanisms when enabling new features. If a test fails, then devices fall back to using MIDI 1.0 for that feature. MIDI-CI improves MIDI capabilities in several key areas.

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

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

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


Key Features of the MIDI 2.0 Environment
Three Ps
Future MIDI Expansion includes three main areas enabled by MIDI Capability Inquiry
Profiles
MIDI 2.0™
Channel Voice Messages
Profile Configuration
Profiles are defined sets of rules for how a MIDI device sends or responds to specific MIDI messages to achieve a specific purpose or suit a specific application.
Property Exchange
Property Exchange (PE) messages can get and set device properties including but not limited to product name, configuration settings, controller names, controller values, patch names and other meta data, etc.
Protocol Negotiation
MIDI-CI defines how to negotiate to MIDI 2.0. Devices that do not support MIDI 2.0 will continue to use MIDI 1.0.
Not official logos


MIDI 2.0 Environment
Use MIDI 1.0 Defaults
Use MIDI 1.0 Defaults
Use MIDI 1.0 Protocol
Profiles Not Supported
Property Exchange Not Supported
Protocol Test Failure
MIDI Device
MIDI-CI
MIDI Device
Profile Configuration
Property Exchange
Protocol Negotiation
Auto Configure:
General MIDI
Piano
Organ
MPE
MIDI Show Control
Etc.
Get, Set, Recall:
Manufacturer & Model
List of Patches
Controller Mappings
Synthesis Parameters
MIDI Implementation
Etc.
MIDI 1.0 Protocol
MIDI 2.0 Protocol

1.2 PROFILE CONFIGURATION

There are some common types of MIDI devices that all tend to do very similar things. We can define Profiles to define how MIDI controls the common features. MIDI-CI Profile Configuration allows devices to discover and turn on Profiles for better interoperability and ease of use while lowering the need for manual configuration of devices by users.

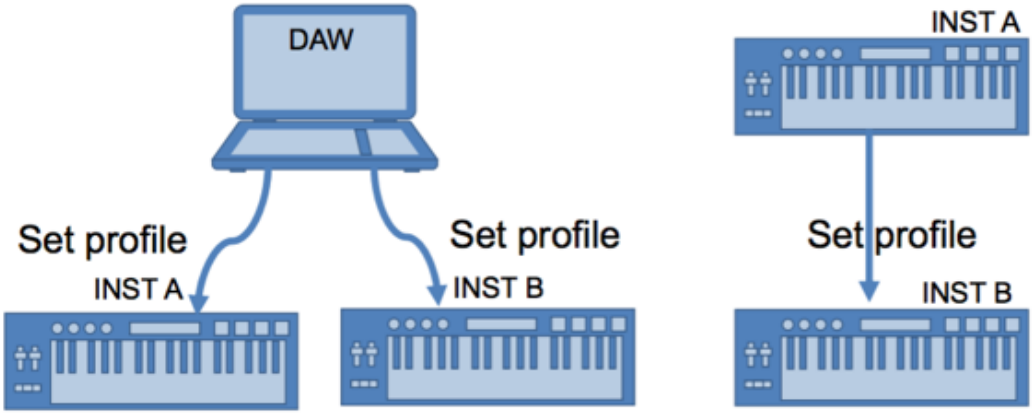
To explain, let's consider MIDI controlled pianos. Pianos have a lot of characteristics in common and we can control those characteristics by a common set of MIDI messages. MIDI messages used by all pianos include Note On/Off and Sustain Pedal. A Piano Profile might define that Note Number 60 is Middle C, define a specific velocity response curve, define the use of a variable Sustain Pedal (not just On/Off) message, define a controller message for the angle of the lid opening, define a message to select the type of stretch tuning, and more. Any device that reports support for the Piano Profile would have to conform to that design.

Advanced MIDI users might be familiar with manually "mapping" all the controllers from one device to another device to make them talk to each other. If 2 devices agree to use a common Profile, MIDI-CI Profile Configuration can auto-configure the mappings. Profiles can be written for device types or for unique applications that are used across multiple device types. Profiles might be written for instruments such as pianos, electric pianos, drawbar organs, drum sets, analog synthesizers. Feature Profiles could define common messages to control orchestral articulation, direct pitch control models, or per-note expression. Profile can also serve non-musical applications such as lighting controllers or industrial machines.

The following video has a demonstration of how Profile Configuration works.

Profile Configuration Demo

- Set a profile on each inst with one App.
- Confirm the compatibility before/after the profile on
- Show the profile will work both on PC-MI and MI-MI



24

1.3 PROPERTY EXCHANGE

Property Exchange is a set of System Exclusive messages that devices can use discover, get, and set many properties of MIDI devices. The properties that can be exchanged include device configuration settings, a list of patches with names and other meta data, a list of controllers and their destinations, and much more.

Property Exchange can allow for devices to auto map controllers, choose programs by name, 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 Windows, Mac, Linux, IOS and Web Browsers and may provide tighter integrations with DAW's and hardware controllers.

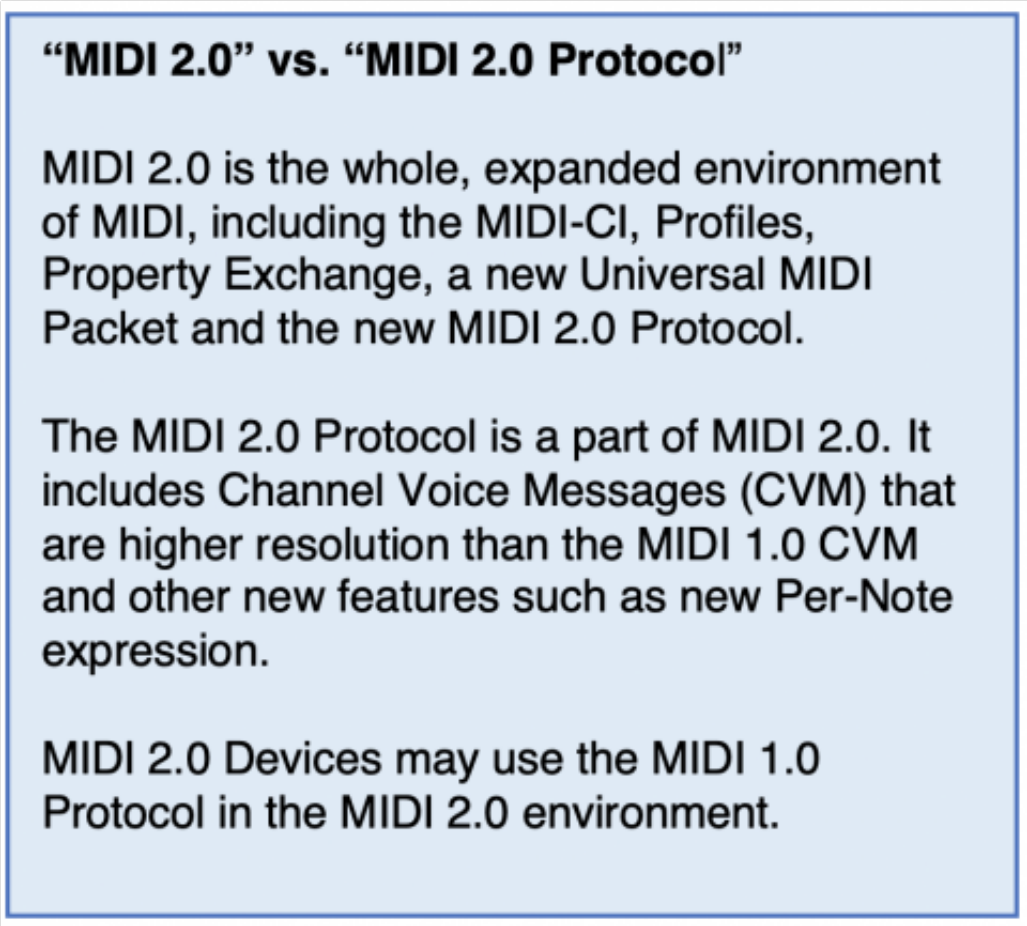
Property Exchange uses JSON inside of the System Exclusive messages. JSON (JavaScript Object Notation) is a human readable format for exchanging data sets. The use of JSON expands MIDI with a whole new area of potential capabilities.

1.4 PROTOCOL NEGOTIATION

MIDI-CI Protocol Negotiation allows devices to select between using the MIDI 1.0 Protocol or the MIDI 2.0 Protocol. Two devices that have established a 2-way MIDI-CI session can select a Protocol and features of that Protocol.

A MIDI Protocol is the language of MIDI, or the set of messages that MIDI uses. Architectural concepts and semantics from MIDI 1.0 are the same in the MIDI 2.0 Protocol. Compatibility for translation to/from MIDI 1.0 Protocol is given high priority in the design of MIDI 2.0 Protocol.

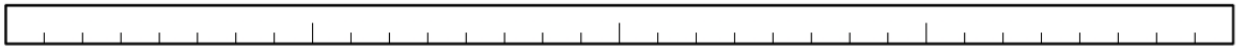
The MIDI 1.0 Protocol and the MIDI 2.0 Protocol have many messages in common, messages that are identical in both protocols. The MIDI 2.0 Protocol extends some MIDI 1.0 messages with higher resolution and new features. There are newly defined messages. Some can be used in both protocols and some are exclusive to the MIDI 2.0 Protocol.



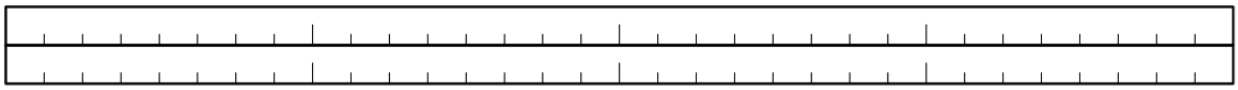
1.5 MIDI 1.0 Protocol, MIDI 2.0 Protocol and the Universal MIDI Packet

MIDI 2.0 has a new Universal MIDI Packet format for carrying MIDI 1.0 Protocol messages and MIDI 2.0 Protocol messages. A Universal MIDI Packet contains a MIDI message which consists of one to four 32-bit words.

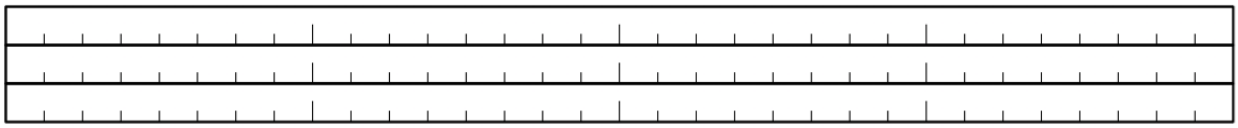
Example 1: 32-Bit Message in a Single 32-Bit Universal MIDI Packet



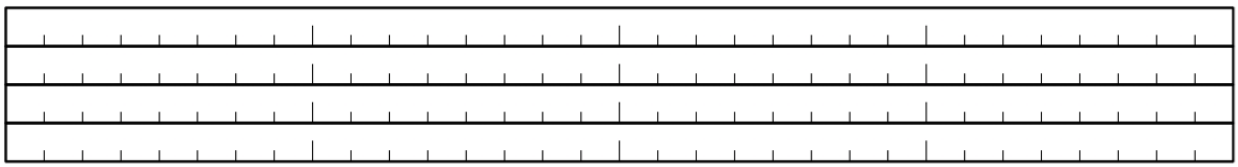
Example 2: 64-Bit Message in a Single 64-Bit Universal MIDI Packet



Example 3: 96-Bit Message in a Single 96-Bit Universal MIDI Packet



Example 4: 128-Bit Message in a Single 128-Bit Universal MIDI Packet



The Universal MIDI Packet format is suited to sending MIDI data over high speed transports such as USB or a network connection or between applications running inside a personal computer OS.

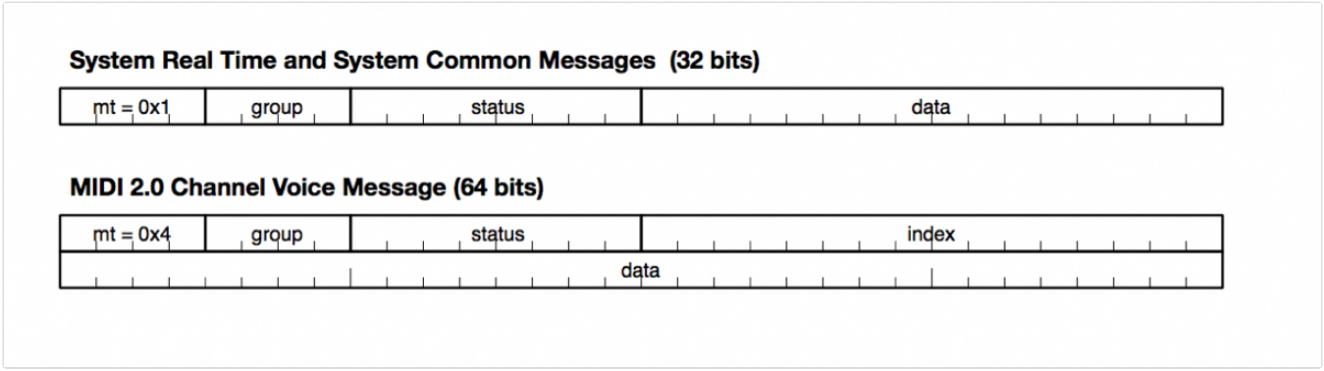
The traditional 5 pin DIN transport from MIDI 1.0 uses a byte stream rather than packets. At the moment, there is no plan to use the Universal MIDI Packet on the 5 pin DIN transport. Unless/Until that plan changes, 5 pin DIN will only support the MIDI 1.0 Protocol.

1.5.1 Message Types

The first 4 bits of every message contain a Message Type. The Message Type is used as a classification of message functions.

MT	Packet Size	Description
0x0	32 bits	Utility Messages
0x1	32 bits	System Real Time and System Common Messages (except System Exclusive)
0x2	32 bits	MIDI 1.0 Channel Voice Messages
0x3	64 bits	Data Messages (including System Exclusive)
0x4	64 bits	MIDI 2.0 Channel Voice Messages
0x5	128 bits	Data Messages
0x6	32 bits	Reserved
0x7	32 bits	Reserved
0x8	64 bits	Reserved
0x9	64 bits	Reserved
0xA	64 bits	Reserved
0xB	96 bits	Reserved
0xC	96 bits	Reserved
0xD	128 bits	Reserved
0xE	128 bits	Reserved
0xF	128 bits	Reserved

Message Type Examples:



1.5.2 Groups

The Universal MIDI Packet carries 16 Groups of MIDI messages, each Group containing an independent set of System Messages and 16 MIDI Channels. Therefore, a single connection using the Universal MIDI Packet carries up to 16 sets of System Messages and up to 256 Channels.

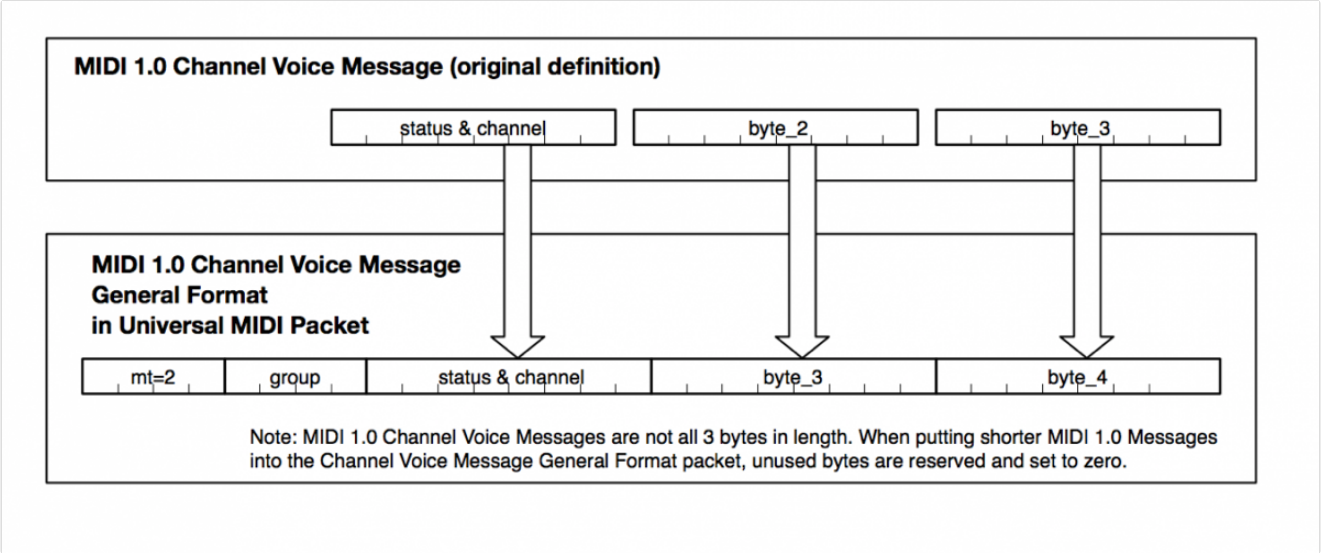
Each of the 16 Groups can carry either MIDI 1.0 Protocol or MIDI Protocol. Therefore, a single connection can carry both protocols simultaneously. MIDI 1.0 Protocol and MIDI Protocol messages cannot be mixed together within 1 Group.

1.5.3 Jitter Reduction Timestamps

The Universal MIDI Packet format adds a Jitter Reduction Timestamp mechanism. A Timestamp can be prepended to any MIDI 1.0 Protocol message or MIDI 2.0 Protocol message for improved timing accuracy.

1.5.4 MIDI 1.0 Protocol Inside the Universal MIDI Packet

All existing MIDI 1.0 messages are carried in the Universal MIDI 1.0. As an example, this diagram from the protocol specification shows how MIDI 1.0 Channel Voice Messages are carried in 32-bit packets:



System messages, other than System Exclusive, are encoded similarly to Channel Voice Messages. System Exclusive messages vary in size, can be very large, and can span multiple Universal MIDI Packets.

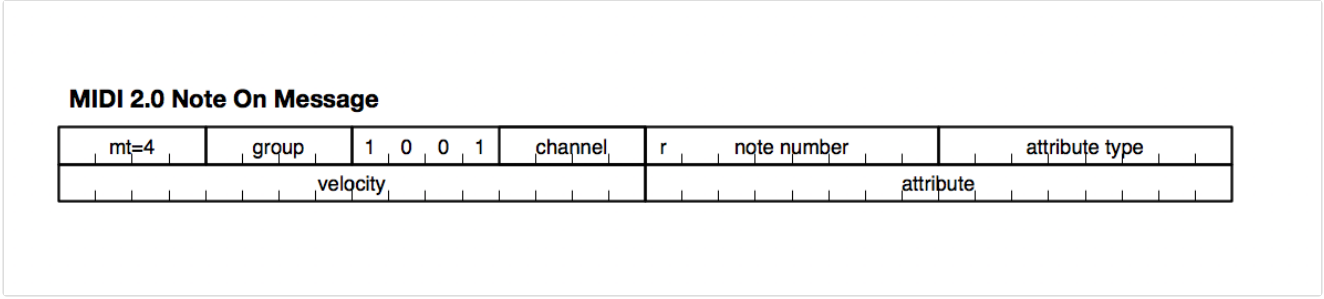
1.5.5 MIDI 2.0 Protocol Messages

The MIDI 2.0 Protocol uses the architecture of MIDI 1.0 Protocol to maintain backward compatibility and easy translation while offering expanded features.

- Extends the data resolution for all Channel Voice Messages.
- Makes some messages easier to use by aggregating combination messages into one atomic message.
- Adds new properties for several Channel Voice Messages.
- Adds several new Channel Voice Messages to provide increased Per-Note control and musical expression.
- Adds 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.
- Keeps all System messages the same as in MIDI 1.0.

Expanded Resolution and Expanded Capabilities

This example of a MIDI 2.0 Protocol Note message shows the expansions beyond the MIDI 1.0 Protocol equivalent. The MIDI 2.0 Protocol Note On has higher resolution Velocity. The 2 new fields, Attribute Type and Attribute data field, provide space for additional data such as articulation or tuning details



Easier to Use: Registered Controllers (RPN) and Assignable Controllers (NRPN)

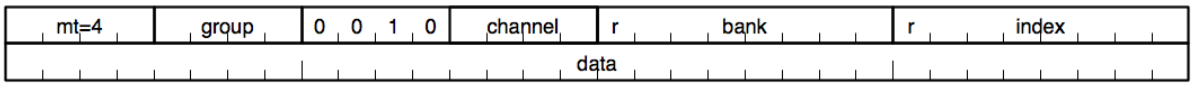
Creating and editing RPNs and NRPNs with MIDI 1.0 Protocol requires the use of compound messages. These can be confusing or difficult for both developers and users. MIDI 2.0 Protocol replaces RPN and NRPN compound messages with single messages. The new Registered Controllers and Assignable Controllers are much easier to use.

The MIDI 2.0 Protocol replaces RPN and NRPN with 16,384 Registered Controllers and 16,384 Assignable Controller that are as easy to use as Control Change messages.

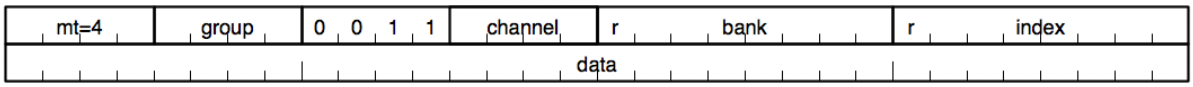
Managing so many controllers might be cumbersome. Therefore, Registered Controllers are organized in 128 Banks, each Bank having 128 controllers. Assignable Controllers are also organized in 128 Banks, each Bank having 128 controllers.

Registered Controllers and Assignable Controllers support data values up to 32bits in resolution.

MIDI 2.0 Registered Controller Message



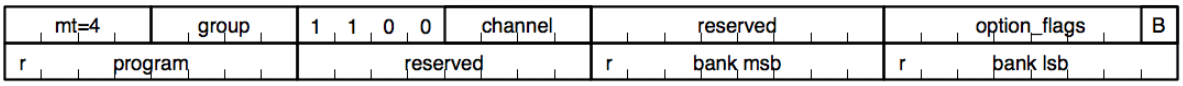
MIDI 2.0 Assignable Controller Message



1.5.6 MIDI 2.0 Program Change Message

MIDI 2.0 Protocol combines the Program Change and Bank Select mechanism from MIDI 1.0 Protocol into one message. The MIDI 1.0 mechanism for selecting Banks and Programs requires sending three MIDI messages. MIDI 2.0 changes the mechanism by replicating the Banks Select and Program Change in one new MIDI 2.0 Program Change message. Banks and Programs in MIDI 2.0 translate directly to Banks and Programs in MIDI 1.0.

MIDI 2.0 Program Change Message



The MIDI 2.0 Program Change message always selects a Program. A Bank Valid bit (B) determines whether a Bank Select is also performed by the message.

If Bank Valid = 0, then the receiver performs the Program Change without selecting a new Bank; The receiver keeps its currently selected Bank. Bank MSB and Bank LSB data fields are filled with zeroes.

If Bank Valid = 1, then the receiver performs both Bank and Program Change.

Other option flags that are not defined yet and are Reserved.

1.5.7 New Data Messages for MIDI 1.0 Protocol and MIDI 2.0 Protocol

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. Both messages can be used when using the Universal MIDI Packet format for MIDI 1.0 Protocol or MIDI 2.0 Protocol.

1.6 The Future of MIDI 1.0

MIDI 1.0 is not being replaced. Rather it is being extended and is expected to continue, well integrated with the new MIDI 2.0 environment. It is part of the Universal MIDI Packet, the fundamental MIDI data format. Many MIDI devices will not need any of the new features of MIDI 2.0 in order to perform all their functions. Some devices will continue to use the MIDI 1.0 Protocol while using other extensions of MIDI 2.0, such as Profile Configuration or Property Exchange.

1.7 What's Next

There is already a USB-IF working group that is working on a MIDI 2.0 USb specification. Meeting at Winter NAMM were held with both the major OS providers (Apple, Google and Microsoft) as well as DAW companies.

Roland has released a MIDI 2.0 ready controller.

On the Monday after NAMM, some of the dedicated team of volunteer MMA and AMEI members held an all day planning meeting to map out the plans for the next year.

In the meantime, MIDI 1.0 works well. In fact, MIDI 2.0 is just more MIDI. As new features arrive on new instruments, they will work with existing devices and system. The same was true for long list of other additions made to MIDI since 1983. MIDI 2.0 is just part of the evolution of MIDI that has gone on for 36 years. The step by step evolution continues.

1.8 Why Join the MMA (MIDI Manufacturers Association)

If you are a developer of MIDI software or hardware, there are a lot of reasons to join the MIDI Manufacturers Association now. This article includes some information about MIDI 2.0, but it is definitely not enough to start developing a MIDI 2.0 product. The reason we do not release specification details before they are finally approved is that if information is released too early and then changes are made, it can lead to interoperability problems.

If you join the MMA now, you not only get access to the current version of the full MIDI 2.0 specification, but will also have a voice in future MIDI specifications including Profiles and Property Exchange messages.

To implement MIDI-CI and MIDI 2.0, you need a manufacturers SysEx ID. A SysEx ID by itself is $260 a year, but it is included with your MMA membership. You will also have access to the MMA Github which has code for MIDI 2.0 to MIDI 1.0 translation (and vis versa), MIDI 2.0 Scope, a tool for sending and testing MIDI 2.0 messages developed by Art and Logic and Property Exchange Work Bench, an application developed by Yamaha for prototyping and testing Property Exchange.

We are also working on a MIDI 2.0 logo and logo licensing program.

So we encourage you to join the MIDI Manufacturers Association now and get access to all the documents you will need to get a head start on MIDI 2.0.

Join the MMA

MIDI 2.0 FAQs

We have been monitoring the comments on a number iof websites and wanted to provide some FAQs about MIDI 2.0 as well as videos of some requested MIDI 2.0 features.

Will MIDI 2.0 devices need to use a new connector or cable?

No, MIDI 2.0 is a transport agnostic protocol.

- Transport- To transfer or convey from one place to another
- Agnostic- designed to be compatible with different devices
- Protocol-a set of conventions governing the treatment and especially the formatting of data in an electronic communications system

That's engineering speak for MIDI 2.0 is a set of messages and those messages are not tied to any particular cable or connector.

When MIDI first started it could only run over the classic 5 Pin DIN cable and the definition of that connector and how it was built was described in the MIDI 1.0 spec.

However soon the MIDI Manufacturers Association and Association of Music Electronic Industries defined how to run MIDI over many different cables and connectors.

So for many years, MIDI 1.0 has been a transport agnostic protocol..

MIDI 1.0 messages currently run over 5 PIN Din, serial ports, Tip Ring Sleeve 1/8" cables, Firewire and Ethernet and all the different variations of USB cables,

Can MIDI 2.0 run over those different MIDI 1.0 transports now?

No, there needs to be new specifications written for each transport. There is a new Universal Packet Format that will be common to all modern transports that will help make this work move quicker. The new Universal Packet contains both MIDI 1 .0 messages and MIDI 2.0 messages plus some messages that can be used with both.

The most popular MIDI transport today is USB. The vast majority of MIDI products are connected to computers or hosts via USB.

USB is the first target for MIDI 2.0.

Can MIDI 2.0 provide more reliable timing?

Yes, and not only that the timing for MIDI 1.0 can also be improved. One of the new messages that can work with both MIDI 1.0 and MIDI 2.0 are Jitter Timestamps.

Goals of JR Timestamps:

- Capture a performance with accurate timing
- Transmit MIDI message with accurate timing over a system that is subject to jitter
- Does not depend on system-wide synchronization, master clock, or explicit clock synchronization between Sender and Receiver.

Note: There are two different sources of error for timing: Jitter (precision) and Latency (sync). *The Jitter Reduction Timestamp mechanism only addresses the errors introduced by jitter. The problem of synchronization or time alignment across multiple devices in a system requires a measurement of latency. This is a complex problem and is not addressed by the JR Timestamping mechanism.*

Can MIDI 2.0 provide more resolution?

Yes, MIDI 1.0 messages are usually 7 bit (14 bit is possible by not widely implmented because there are only 128 CC messages). In MIDI 2.0 velocity is 16 bit and the 128 control change messages, 16,384 Registered Controllers, 16,384 Assignable Controllers, Poly and channel pressure and Pitch Bend are 32 bit.

Can MIDI 2.0 make it easier to have microtonal control and different non-western scales?

Yes, MIDI 2.0 allows direct pitch control of notes (see videos)