MIDI-CI Property Exchange Controller Resources

AllCtrlList, ChCtrlList
CtrlMapList

MIDI Association Document: M2-117-UM

Document Version 1.0
Draft Date May 19, 2023

Published June 15, 2023

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

PREFACE

MIDI Association Document M2-117-UM
MIDI-CI Property Exchange Controller Resources

Property Exchange is part of the MIDI-CI specification. Property Exchange is a method for sending JSON over System Exclusive to exchange data between two devices. This document describes the Property Data for AllCtrlList, ChCtrlList and CtrlMapList Resources. For information on how to transmit and receive payloads over System Exclusive please read the MIDI-CI and Common Rules for MIDI-CI Property Exchange specifications.

http://www.amei.or.jp

https://www.midi.org

Version History

Table 1 Version History

Publication Date	Version	Changes
June 15, 2023	1.0	Initial Version

Contents

Figures

Tables

1 Introduction

1.1 Executive Summary

The Property Exchange Resources described in this document allow an Initiator to discover the Controller messages a device recognizes and transmits.

Examples of how this may be used:

- A DAW communicates with Device and adds supported controllers to the automation window of a MIDI track
- A Keyboard communicates with a MIDI Device, such a tone generator or soft synth, and automatically assigns its physical controls to the most appropriate Controller Messages.
- A DAW acting as a proxy between a MIDI Device and a Keyboard to provide autoconfiguration for the user.

1.2 Background

Property Exchange is part of the MIDI Capability Inquiry (MIDI-CI) ***[MA03]*** specification and MIDI 2.0. Property Exchange is a method for getting and setting various data, called Resources, between two Devices. Resources are exchanged inside two payload fields of System Exclusive Messages defined by MIDI-CI, the Header Data field and Property Data field. This document defines only the contents of the Header Data and Property Data fields. For information on how to transmit and receive these Resource payloads inside MIDI-CI System Exclusive messages, please see the MIDI Capability Inquiry specification ***[MA03]*** and Common Rules for MIDI-CI Property Exchange specification ***[MA05]***.

1.3 References

1.3.1 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 1.2, Association of Musical Electronics Industry, http://www.amei.or.jp/, and The MIDI Association, https://www.midi.org/

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

[MA05] ***M2-103-UM Common Rules for Property Exchange***, Version 1.0, 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] ***M2-115-U MIDI 2.0 Bit Scaling and Resolution***, Version 1.0, Association of Musical Electronics Industry, http://www.amei.or.jp/, and The MIDI Association, https://www.midi.org/

[MA08] ***M2-113-UM Default Control Change Mapping Profile***, Version 1.0, Association of Musical Electronics Industry, http://www.amei.or.jp/, and The MIDI Association, https://www.midi.org/

[MA09] ***M2-105-UM MIDI-CI Property Exchange Foundational Resources***, 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

No informative references.

1.4 Terminology

1.4.1 Definitions

Active Controller Message: Any of the Controller Messages which a Device is currently recognizing or transmitting on a selected channel.

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

Controller Message: Any MIDI Message from the following list:

MIDI 1.0 and MIDI 2.0 Protocol:

- Control Change
- Channel Pressure (Aftertouch)
- Poly Pressure (Key Aftertouch)
- Registered Controller (RPN)
- Assignable Controller (NRPN)
- Pitch Bend

MIDI 2.0 Protocol only:

- Per-note Registered Controller (including Relative versions)
- Per Note Assignable Controller (including Relative versions)
- Per Note Pitch Bend

DAW: Digital Audio Workstation

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

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]***.

List Resource: A specific type of Resource that provides a list of objects in a JSON array.

MA: MIDI Association.

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 ***[MA07]***.

MIDI-CI: MIDI Capability Inquiry ***[MA03]***, 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-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 Association: Authority for MIDI specifications worldwide except Japan. See also MMA.

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

MMA: MIDI Manufacturers Association.

PE: Property Exchange.

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]***.

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.

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]***.

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]***.

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).

1.5 Data Format, Bit Scaling and Resolution

All values in Resource defined in this Controller Resources specification are presented as 32bit values. These 32bit values are the data values contained in the associated MIDI 2.0 Protocol Controller messages.

For critical information on understanding resolution of Property Values, see the MIDI 2.0 Bit Scaling and Resolution ***[MA07]*** specification. That document defines recommended practices for scaling values, handling of stepped/enumerated values and translating values between MIDI 1.0 Protocol and MIDI 2.0 Protocol.

2 Resource: AllCtrlList

AllCtrlList Resource is the current set of Active Controller Messages of the Responder on all currently-active channels.

Each entry in the Property data of the List Resource provides useful information such as the default value, acceptable ranges and if the Active Controller Message is transmit and recognize (default) or if it is transmit or recognize only.

2.1 Initiator Requests Data from a Responder Using an Inquiry: Get Property Data

An Initiator may request the **AllCtrlList** Resource from a Responder using an **Inquiry: Get Property Data message**.

Table 4 Initiator Sends Inquiry: Get Property Data

`Header Data`	`{"resource":"AllCtrlList"}`
`Property Data`	*None*

2.1.1 Responder Sends Reply to Inquiry Get Property Data

The reply is an array of Active Controller Message entries.

Each entry may be assigned a priority level, by including a "**`priority`**" Property to highlight the most useful Active Controller Messages. A hardware controller with a limited number of sliders and knobs can use the "**`priority`**" Property to easily auto assign the most useful Active Controller Messages to these physical controls.

Results should be listed in order by the value of the "**`priority`**" Property.

Supported Active Controller Messages shall not be listed more than once, even if the controller is routed to multiple destinations except where the transmit and recognize Active Controller Message differs. See Section 2.1.3 for more information.

Table 5 Controller Message Data Object

Property Key	Property Value Type	Description
`title`	string, required	Human-readable name of the Active Controller Message, pertaining to the effect of recognized messages or the physical interface generating messages. This may be displayed to the user by a connected device.
`description`	string, commonmark [COMM01]	Human-readable description of the effect of recognized messages or the physical interface generating messages. This may be displayed to the user by a connected device.
`channel`	number (integer) (1-256, required)	This is the Channel for the Controller being described by this entry in the list of Controller Message Data Object. Channel numbering starts on the first channel on the first Group in the Function Block and continues across Groups in the Function Block. For example, if a Function Block contains 3 Groups it has Channels 1-48
`ctrlType`	enum, required – "cc", "chPress", "pPress", "nrpn", "rpn", "pBend", "pnrc", "pnac", "pnp"	Type of Controller Message. MIDI 1.0 and MIDI 2.0 protocol: • cc - Control Change • chPress- Channel Pressure (Aftertouch) • pPress- Poly Pressure (Key Aftertouch)

		• rpn - Registered Controller (RPN) • nrpn - Assignable Controller (NRPN) • pBend- Pitch Bend MIDI 2.0 protocol only: • pnrc - Per-note Registered Controller • pnac - Per Note Assignable Controller • pnp - Per Note Pitch Bend
`ctrlIndex`	array of integers, required*	This is an array of the Controller Message index numbers. For (N)RPN this will be the Bank (MSB) and Index (LSB). For CC this would mostly be just a single item for the CC, however this can be a pair of MSB and LSB to have 14bit CC messages.
`priority`	integer (1-5)	A value of 1 is the most important, descending down to 5 as the least important. When entries are sent in the **AllCtrlList** Resource array, they should be ordered by the "**`priority`**" Property. Initiators should use the ordering of the **AllCtrlList** Resource to complement the priority given. The Initiator may decide the best way to display or handle entries based on priority
`default`	integer	The default value if entry is reset. Values declared in unsigned 32 bit values
`transmit`	enum "absolute", "relative", "both", "none" default: "absolute"	Declares whether this entry supports absolute values, relative values using Relative Registered/Assignable Controllers, both, or none. If not given, the Device only transmits using absolute values.
`recognize`	enum "absolute", "relative", "both", "none" default: "absolute"	Declares whether this entry supports absolute values, relative values using Relative Registered/Assignable Controllers, both, or none. If not given, the Device only recognizes using absolute values.
`numSigBits`	integer	Number of significant bits used for the 32 bit values. 32, if not provided. Depending on the current protocol this number might be larger than the maximum number of bits (e.g. 7 bits for controller in MIDI 1 protocol). Refer to the MIDI 2.0 Bit Scaling and Resolution ***[MA07]*** specification.
`paramPath`	string (JSON Pointer) Max length 256 bytes	The Parameter Path indicates the grouping and display of Active Controller Messages, e.g. `/Oscillator/A/Envelope/Attack` This allows the Initiator to keep related Controls together when displaying them in a model tree or similar.
`typeHint`	enum "continuous", "momentary", "toggle", "relative", "valueSelect"	Provides a hint as to what physical (or displayed) control mechanism should be used on the Initiator to generate the Active Controller Messages.

ctrlMapId	string (max 36 chars, "a-z", "0-9" or "_" characters only)	This Property provides the "**resId**" that shall be used with a **CtrlMapList** Resource. Subsequently, the **CtrlMapList** Resource provides a set of enumerated values for the Active Controller Message (see Section 3).
stepCount	integer	If Active Controller Message only supports a discrete number of steps, set this value to declare the number of steps. Refer to Stepped Values and Enumerations in the MIDI 2.0 Bit Scaling and Resolution ***[MA07]*** specification.
minMax	array of 2 integers	When a Responder recognizes an Active Controller Message within a limited range of the possible values, it may declare the minimum and maximum values recognized. Values declared in unsigned 32 bit values.

* This is not required when "**ctrlType**" is set to "chPress", "pPress", "pBend", "pnp"

2.1.1.1 "priority" Property

This value indicates the importance or prominence of the entry for purposes of display in a GUI or automatic mapping in a peer Device.

1. Essential. The Active Controller Message should be exposed to users whenever possible, even to the exclusion of other functionality.
2. High. The Active Controller Message should typically be exposed to users.
3. Medium. The Active Controller Message may or may not be exposed to users by default.
4. Low. The Active Controller Message should typically not be exposed to users by default.
5. Hidden. The Active Controller Message represents an implementation detail or internal function that should not be exposed to end users.

Devices may have a limited capacity for essential Active Controller Messages. For this reason, the list of Essential Active Controller Messages should be as small as possible — for example, many Devices have eight to ten faders. When a Device is not able to accommodate all items within a priority class, items appearing earlier in the list may be exposed with preference.

2.1.1.2 "typeHint" Property

The "momentary" and "toggle" Type Hints should send only 2 values for on/off, yes/no, etc.

- "momentary" - A binary switch where each time the value is set to on/yes, the receiver property is set to on and when the switch is set to off/no the receiver property is set to off/no. (Example: Sustain Pedal)
- "toggle" - A binary switch where each time the value changes from off/no to on/yes, the receiver alternates between on/yes and off/no.

If "**typeHint**" Property Value is "valueSelect" then "**ctrlMapId**" Property shall be set. The "valueSelect" Type should send only specific values as declared from the **CtrlMapList** Property Data returned using the Property Value. See Section 3.

The "continuous" Type should send a contiguous set of values between the minimum and maximum values declared in the "**minMax**" Property.

Continuous types can be any control that represents a

- slider
- dial
- ribbon
- xAxis
- yAxis

- zAxis
- spread
- angle
- intensity
- azimuth
- distance

The "`relative`" Type should send relative values. Examples of relative controls include:

- encoder
- up down buttons

2.1.1.3 "`minMax`" Property

The "**`minMax`**" Property declares the recognized range of the Active Controller Message. A transmitting Device may send Active Controller Messages outside of this range; however, these messages may be ignored or could produce unexpected results.

Manufacturers are encouraged to use the full range and use the "**`minMax`**" Property for describing legacy products. Manufacturers are instead encouraged to use the method detailed in the "**`stepCount`**" Property.

"**`minMax`**" and "**`stepCount`**" shall not be used on the same **AllCtrlList** Property Data entry.

2.1.2 Example of a AllCtrlList Reply to Get Property Data Message

Table 6 Responder Sends Reply to Get Property Data

Header Data	{"status":200}
Property Data	[{ "title": "Volume", "ctrlType": "cc", "channel": 1, "default": 4294967295, "paramPath": "/volume", "ctrlIndex": [7,39], "ctrlMapId": "volumeDb" }, { "title": "Filter Cutoff", "ctrlType": "nrpn", "channel": 2, "default": 214748364864, "paramPath": "/filter/cutoff", "ctrlIndex": [0,55], "recognize": "both", "ctrlMapId": "freq" }, { "title": "Hold Pedal",

<table>
<tr><td></td><td><pre>
        "ctrlType": "cc",
        "channel": 1,
        "default": 0,
        "ctrlIndex": [67]
        "typeHint": "momentary
    },
    {
        "title": "LFO Wave type",
        "ctrlType": "cc",
        "channel": 2,
        "default": 0,
        "paramPath": "/lfo/waveType",
        "ctrlIndex": [14],
        "ctrlMapId": "lfoWaveType",
        "typeHint": "valueSelect"
    }
]
</pre></td></tr>
</table>

2.1.3 Separate Recognize and Transmit on the same Active Controller Message

Responders which both transmit and recognize the same Active Controller Message index may optionally send two Active Controller Message Data Objects for the same "**ctrlIndex**", describing the physical interface generating the Active Controller Message and the effect of recognized messages.

The only time a "**ctrlType**"-"**ctrlIndex**"-**"channel"** combination for a given Active Controller Message may be listed twice is when one specifies transmit behavior and the other specifies recognize behavior.

Table 7 Responder Sends Reply to Get Property Data

<table>
<tr><td>Header Data</td><td><pre>{"status":200}</pre></td></tr>
<tr><td>Property Data</td><td><pre>
[
    {
        "title": "Vibrato",
        "ctrlType": "cc",
        "channel": 1,
        "default": 0,
        "ctrlIndex": [2],
        "transmit": "none",
    },
    {
        "title": "Modulation Wheel",
        "ctrlType": "cc",
        "channel": 1,
        "default": 0,
        "ctrlIndex": [2],
        "recognize": "none"
    },
</pre></td></tr>
</table>

	]

2.2 ResourceList Integration for AllCtrlList

Table 8 Minimal entry in ResourceList

Property Data	[{"resource": "AllCtrlList"}]

Table 9 Full version with default settings

Property Data	[
	{
	"resource": "AllCtrlList",
	"canGet": true,
	"canSet": "none",
	"canSubscribe": false,
	"schema": {
	"type": "array",
	"title": "Active Controller Messages List",
	"$ref": "http://schema.midi.org/property-exchange/M2-117-S_v1-0_AllCtrlList.json"
	},
	"columns": [
	{"property": "title"},
	{"property": "priority"},
	{"property": "ctrlType"}
	]
	}
	]

2.3 Integration of AllCtrlList with MIDI Message Report

MIDI-CI Process Inquiry describes how to retrieve Active Controller Message values using an Inquiry: MIDI Message Report message. See *[MA03]*.

It is recommended that Devices that support **AllCtrlList** Resource also support **MIDI-CI Inquiry: MIDI Message Report**. An Initiator should use this message after retrieving a **AllCtrlList** Property Data and use the "**`ctrlType`**", "**`ctrlIndex`**", and **`"channel"`** Properties to match the incoming MIDI Messages to the **AllCtrlList** Property Data entries.

3 Resource: ChCtrlList

ChCtrlList Resource is the current set of Active Controller Messages of the Responder for a Channel Entry as described in **ChannelList** Resource. This Resource cannot be used without a reference from the **ChannelList** Resource. See **ChannelList** Resource in MIDI-CI Property Exchange Foundational Resources ***[MA09]***.

Each entry in the Property data of the List Resource provides useful information such as the default value, acceptable ranges and if the Active Controller Message is transmit and recognize (default) or if it is transmit or recognize only.

3.1 Initiator Requests Data from a Responder Using an Inquiry: Get Property Data

An Initiator may request the **ChCtrlList** Resource from a Responder using an **Inquiry: Get Property Data message**.

Table 10 Initiator Sends Inquiry: Get Property Data

`Header Data`	`{"resource":"ChCtrlList"}`
`Property Data`	*none*

3.1.1 Responder Sends Reply to Inquiry Get Property Data

The reply is an array of Active Controller Message entries.

Each entry may be assigned a priority level, by including a "**`priority`**" Property to highlight the most useful Active Controller Messages. A hardware controller with a limited number of sliders and knobs can use the "**`priority`**" Property to easily auto assign the most useful Active Controller Messages to these physical controls.

Results should be listed in order by the value of the "**`priority`**" Property.

Supported Active Controller Messages shall not be listed more than once, even if the controller is routed to multiple destinations except where the transmit and recognize Active Controller Message differs. See Section 2.1.3 for more information.

Table 11 Controller Message Data Object

Property Key	Property Value Type	Description
`Title`	string, required	Human-readable name of the Active Controller Message, pertaining to the effect of recognized messages or the physical interface generating messages. This may be displayed to the user by a connected device.
`description`	string, commonmark [COMM01]	Human-readable description of the effect of recognized messages or the physical interface generating messages. This may be displayed to the user by a connected device.
`ctrlType`	enum, required – "cc", "chPress", "pPress", "nrpn", "rpn", "pBend", "pnrc", "pnac", "pnp"	Type of Controller Message. MIDI 1.0 and MIDI 2.0 protocol: • cc - Control Change • chPress- Channel Pressure (Aftertouch) • pPress- Poly Pressure (Key Aftertouch) • rpn - Registered Controller (RPN) • nrpn - Assignable Controller (NRPN) • pBend- Pitch Bend MIDI 2.0 protocol only:

		• pnrc - Per-note Registered Controller • pnac - Per Note Assignable Controller • pnp - Per Note Pitch Bend
`ctrlIndex`	array of integers, required*	This is an array of the Controller Message index numbers. For (N)RPN this will be the Bank (MSB) and Index (LSB). For CC this would mostly be just a single item for the CC, however this can be a pair of MSB and LSB to have 14bit CC messages.
`Priority`	integer (1-5)	A value of 1 is the most important, descending down to 5 as the least important. When entries are sent in the **ChCtrlList** Resource array, they should be ordered by the "**`priority`**" Property. Initiators should use the ordering of the **ChCtrlList** Resource to complement the priority given. The Initiator may decide the best way to display or handle entries based on priority
`Default`	integer	The default value if entry is reset. Values declared in unsigned 32 bit values
`Transmit`	enum "absolute", "relative", "both", "none" default: "absolute"	Declares whether this entry supports absolute values, relative values using Relative Registered/Assignable Controllers, both, or none. If not given, the Device only transmits using absolute values.
`recognize`	enum "absolute", "relative", "both", "none" default: "absolute"	Declares whether this entry supports absolute values, relative values using Relative Registered/Assignable Controllers, both, or none. If not given, the Device only recognizes using absolute values
`numSigBits`	integer	Number of significant bits used for the 32 bit values. 32, if not provided. Depending on the current protocol this number might be larger than the maximum number of bits (e.g. 7 bits for controller in MIDI 1 protocol). Refer to the MIDI 2.0 Bit Scaling and Resolution ***[MA07]*** specification.
`paramPath`	string (JSON Pointer) Max length 256 bytes	The Parameter Path indicates the grouping and display of Active Controller Messages, e.g. `/Oscillator/A/Envelope/Attack` This allows the Initiator to keep related Controls together when displaying them in a model tree or similar.
`typeHint`	enum "continuous", "momentary", "toggle", "relative", "valueSelect"	Provides a hint as to what physical (or displayed) control mechanism should be used on the Initiator to generate the Active Controller Messages.
`ctrlMapId`	string (max 36 chars, "a-z", "0-9" or "_" characters only)	This Property provides the "**`resId`**" that shall be used with a **CtrlMapList** Resource. Subsequently, the **CtrlMapList** Resource provides a set of enumerated values for the Active Controller Message (see Section 3).
`stepCount`	integer	If Active Controller Message only supports a discrete number of steps, set this value to declare the number of steps. Refer

		to Stepped Values and Enumerations in the MIDI 2.0 Bit Scaling and Resolution ***[MA07]*** specification.
minMax	array of 2 integers	When a Responder recognizes an Active Controller Message within a limited range of the possible values, it may declare the minimum and maximum values recognized. Values declared in unsigned 32 bit values.
defaultCCMap	boolean default: false	Control Change implementation conforms to the Default Control Change Mapping Profile ***[MA08]***, Level 1 (only applicable to Control Change).

* This is not required when "**ctrlType**" is set to "chPress", "pPress", "pBend", "pnp"

3.1.2 Example of a ChCtrlList Reply to Get Property Data Message

Table 12 Responder Sends Reply to Get Property Data

Header Data	{"status":200}

Property Data	[
	{
	"title": "Volume",
	"ctrlType": "cc",
	"default": 4294967295,
	"paramPath": "/volume",
	"ctrlIndex": [7,39],
	"ctrlMapId": "volumeDb",
	"defaultCCMap": true
	},
	{
	"title": "Aftertouch",
	"ctrlType": "chPress",
	"default": 0,
	"transmit": "none"
	},
	{
	"title": "Vibrato",
	"ctrlType": "pnac",
	"default": 2147483648,
	"ctrlIndex": [74]
	},
	{
	"title": "Hold Pedal",
	"ctrlType": "cc",
	"default": 0,
	"ctrlIndex": [67]
	"typeHint": "momentary",
	"defaultCCMap": true
	}

	]

3.1.3 Separate Recognize and Transmit on the same Active Controller Message

Responders which both transmit and recognize the same Active Controller Message index may optionally send two Active Controller Message Data Objects for the same "**ctrlIndex**", describing the physical interface generating the Active Controller Message and the effect of recognized messages.

The only time a "**ctrlType**"-"**ctrlIndex**" pair for a given Active Controller Message may be listed twice is when one specifies transmit behavior and the other specifies recognize behavior.

Table 13 Responder Sends Reply to Get Property Data

Header Data	{"status":200}
Property Data	[{ "title": "Vibrato", "ctrlType": "cc", "default": 0, "ctrlIndex": [2], "transmit": "none", }, { "title": "Modulation Wheel", "ctrlType": "cc", "default": 0, "ctrlIndex": [2], "recognize": "none", "defaultCCMap": true, },]

3.2 ResourceList Integration for ChCtrlList

Table 14 Minimal entry in ResourceList

Property Data	[{"resource": "ChCtrlList"}]

Table 15 Full version with default settings

Property Data	[{ "resource": "ChCtrlList", "canGet": true, "canSet": "none",

	<pre>"canSubscribe": false,
	"requireId": true,
	"schema": {
	"type": "array",
	"title": "Active Controller Messages List for a Channel in ChannelList Resource",
	"$ref": "http://schema.midi.org/property-exchange/M2-117-S_v1-0_ChCtrlList.json"
	},
	"columns": [
	{"property": "title"},
	{"property": "priority"},
	{"property": "ctrlType"}
	]
	}
	]</pre>

3.3 Integration of ChCtrlList with MIDI Message Report

The MIDI-CI MIDI Implementation *[MA05]* describes how to retrieve Active Controller Message values using an MIDI-CI Inquiry: MIDI Message Report message.

It is recommended that Devices that support **ChCtrlList** Resource also support **MIDI-CI Inquiry: MIDI Message Report**. An Initiator should use this message after retrieving a **ChCtrlList** Property Data and use the "**ctrlType**" and, "**ctrlIndex**" Properties to match the incoming MIDI Messages to the **ChCtrlList** Property Data entries.

3.4 Example Integration of ChCtrlList with ChannelList Resource

The **ChCtrlList** Resource is commonly provided as a linked Resource from **ChannelList** Resource data (See MIDI-CI Property Exchange Foundational Resources *[MA09])*.

Table 16 Example ChannelList Property Data with a ChCtrlList Resource in a Link

Property Data	<pre>[
	{
	"title": "Piano + Strings",
	"channel": 1,
	"bankpc": [0,0,76],
	"links": [
	{"resource": "ProgramList", "resId": "general"},
	{"resource": "ChCtrlList", "resId": "ch1"}
	]
	},
	{
	"title": "Rock Drums",
	"channel": 10,
	"bankpc": [50,0,20],
	"links": [</pre>

	{"resource": "ProgramList", "resId": "drums"}, **{"resource": "ChCtrlList", "resId": "ch10"}**] }]

The "**resId**" used for **ChCtrlList** Resource request is provided from the "**link**" property in a **ChannelList** Resource response.

3.4.1 Declaring ChCtrlList for ChannelList entries using Cluster of Channels

ChannelList Resource entries may declare the use of a Cluster of Channels (See MIDI-CI Property Exchange Foundational Resources *[MA09]*). To declare that the Linked **ChCtrlList** Resource is for Cluster Channels the Linked Resource shall contain a **"role"** Property with the value of **"cluster"**.

Table 17 Example ChList Property Data with ChCtrlList Resources and Cluster Channels ChCtrlList Resource in Links

Header Data	{"status":200}

Property Data	[
	{
	"title": "MPE Lower Zone Mode 3",
	"channel": 1,
	"clusterChannelStart": 2,
	"clusterLength": 6,
	"clusterType": "mpe1",
	"programTitle": "MPE Synth",
	"bankPC": [2,0,1],
	"links": [
	{"resource": "ProgramList", "resId": "general"},
	{"resource": "ChCtrlList", "resId": "managerCtrls"}
	{
	"resource": "ChCtrlList",
	"resId": "memberCtrls",
	"role": "cluster"
	}
	]
	}
	]

4 Resource: CtrlMapList

CtrlMapList returns the current set of enumerated values of an Active Controller Message within the range of values recognized by the Responder.

Responders may use **CtrlMapList** to declare the unique meanings for a set of specific values of an Active Controller Message. These values may be indicative of the various points on a control. For example:

- Left, Center, and Right values of a Pan Control Message
- Frequency at specific points throughout the range of a Filter Cutoff Control Message
- Decibel levels at specific points throughout the range of a Volume Control Message
- A set of five LFO Waveforms, using Stepped Values. See ***[MA07]***

This Resource does not provide any information about any Active Controller Message values that are not declared in the Property Data. To determine the possible response of the total range of values, use the **AllCtrlList** and **ChCtrlList** Resource.

4.1 Initiator Requests Data from a Responder Using an Inquiry: Get Property Data

An Initiator may request the **CtrlMapList** Resource from a Responder using an Inquiry: Get Property Data message.

Table 18 Initiator Sends Inquiry: Get Property Data

Header Data	{"resource":"CtrlMapList"}
Property Data	*none*

4.1.1 Responder Sends Reply to Inquiry Get Property Data

The response is an array of Controller Map Data entries.

Table 19 Controller Map Data Object

Property Key	Property Value Type	Description
value	integer, required	An unsigned 32 bit value
title	string, required	Human-readable name of the value.

Table 20 Responder Sends Reply to Get Property Data

Header Data	{"status":200}
Property Data	[{"value": 0, title: "-infinity dB"}, {"value": 167772160, "title": "-100dB"}, {"value": 536870912, "title": "-70dB"}, {"value": 1073741824, "title": "-50dB"}, {"value": 1610612736, "title": "-40dB"}, {"value": 2181570690, "title": "-30dB"}, {"value": 2590615194, "title": "-20dB"}, {"value": 2931485614, "title": "-10dB"}, {"value": 3613226455, "title": "0dB"}, {"value": 4294967295, "title": "+10dB"}]

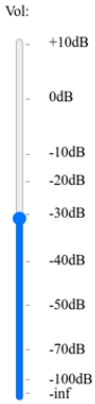

Figure 1 Example of a Device displaying a volume control with the decibel levels.

4.1.2 Example of a Filter Cutoff with Labels

Table 21 Initiator Sends Inquiry: Get Property Data

Header Data	{"resource":"CtrlMapList","resId":"freq"}
Property Data	*none*

Table 22 Responder Sends Reply to Get Property Data

Header Data	{"status":200}
Property Data	[{"value": 0, "title": "20Hz"}, {"value": 1073741824, "title": "100Hz"}, {"value": 2147483648, "title": "800Hz"}, {"value": 3238268993, "title": "7.1Khz"}, {"value": 4294967295, "title": "20Khz"}]

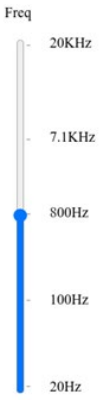

Figure 2 Example of a Device displaying the frequency control with the Hz levels.

4.1.3 Example of a LFO Wave Type Selection

Table 23 Initiator Sends Inquiry: Get Property Data

Header Data	`{"resource":"CtrlMapList","resId":"lfoWaveType"}`
Property Data	*none*

Table 24 Responder Sends Reply to Get Property Data

Header Data	`{"status":200}`

Property Data	`[`
	`{"value": 0, "title": "Sine"},`
	`{"value": 1073741824, "title": "Triangle"},`
	`{"value": 2147483648, "title": "Saw"},`
	`{"value": 3238268993, "title": "Square"}`
	`]`

The LFO Wave Type Control in Section 2.2.2 is described as having a "**`typeHint`**" Property with a value of "`valueSelect`". This could result in a Device being able to display a LFO Wave Type control as a button that cycles through the given values.

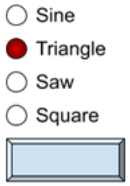

Figure 3 Example of a Device using a button to cycle through the options

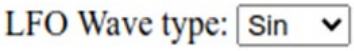

Figure 4 Example of a Device presenting the options as dropdown box

4.2 ResourceList Integration for CtrlMapList

Table 25 Minimal entry in ResourceList

Property Data	`[`
	`{"resource": "CtrlMapList"}`
	`]`

Table 26 Full version with default settings

Property Data	[
	{
	"resource": "CtrlMapList",
	"canGet": true,
	"canSet": "none",
	"canSubscribe": false,
	"requireId": true,
	"schema": {
	"type": "array",
	"title": "Active Controller Map List",
	"$ref": "http://schema.midi.org/property-exchange/M2-117-S_v1-0_CtrlMapList.json"
	},
	"columns": [
	{"property": "title"},
	{"property": "value"}
	]
	}
	]

http://www.amei.or.jp

https://www.midi.org