MIDI-CI Property Exchange Get and Set Device State

StateList Resource

State Resource

Version 1.0

November 24, 2020

Document M2-111-UM

Published By:

Association of Musical Electronics Industry

http://www.amei.or.jp

and

The MIDI Association

https://www.midi.org

MIDI™

PREFACE

Property Exchange is part of the MIDI-CI specifications first released in 2018. Property Exchange is a method for sending JSON over SysEx between two devices to get and set device properties. Each MIDI device is unique and provides an experience different from another device. Property Exchange allows you to discover and use almost any device in a consistent way. This document describes the Property Data for these Resources. For information on how to transmit and receive Property Data over SysEx please see the MIDI-CI [MMA02] and Common Rules for MIDI-CI Property Exchange [MMA03].

Table of Contents

1. Introduction

1.1 Background

Property Exchange is part of the MIDI Capability Inquiry (MIDI-CI) [MMA02] 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 [MMA02] and Common Rules for MIDI-CI Property Exchange specification [MMA03].

The Property Exchange Resources described in this document allow for an Initiator to send or receive Device State, or in other words, to capture a snapshot which might be sent back to the Device at a later time. The primary goal of this application of Property Exchange is to GET the current memory of a MIDI Device. This allows a Digital Audio Workstation (DAW) or other Initiator to store the State of a Responder Device between closing and opening of a project. Before a DAW closes a project, it performs the GET inquiry and the target Device sends a REPLY with all data necessary to restore the current State at a later time. When the DAW reopens a project, the target Device can be restored to its prior State by sending an Inquiry: Set Property Data Message.

Data included in each State is decided by the manufacturer but typically might include the following properties (not an exhaustive list):

Current Program
All Program Parameters
Mode: Single Patch, Multi, etc.
Current Active MIDI Channel(s)
Controller Mappings
Samples and other binary data
Effects
Output Assignments

1.1.1 Two Property Exchange Resources: StateList and State

The StateList Resource provides a list of States of memory regions in a Responder that an Initiator may wish to capture. Following a StateList Resource transaction, the Initiator might use the State Resource mechanism to Get or Set a particular State of a memory region that is included as an object in the StateList Property Data.

1.2 Related Documents

[MMA01] ***The Complete MIDI 1.0 Detailed Specification, Document Version 96.1, Third Edition***, Association of Musical Electronics Industry, http://www.amei.or.jp/, and MIDI Manufacturers Association, https://www.midi.org/.

[MMA02] ***M2-101-UM MIDI Capability Inquiry (MIDI-CI), Version 1.1***, Association of Musical Electronics Industry, http://www.amei.or.jp/, and MIDI Manufacturers Association, https://www.midi.org/.

[MMA03] ***M2-103-UM Common Rules for MIDI-CI Property Exchange, Version 1.1***, Association of Musical Electronics Industry, http://www.amei.or.jp/, and MIDI Manufacturers Association, https://www.midi.org/.

[MMA04] ***M2-105-UM MIDI-CI Property Exchange Foundational Resources: DeviceInfo, ChannelList, JSONSchema, Version 1.0***, Association of Musical Electronics Industry, http://www.amei.or.jp/, and MIDI Manufacturers Association, https://www.midi.org/..

1.3 Terminology

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

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

PE: Property Exchange.

Property: A JSON key:value pair used by Property Exchange.

Property Data: A set of one or more Properties in a Device which are accessible by Property Exchange. Contained in the Property Data field of a MIDI-CI Property Exchange message.

Property Exchange: An AMEI/MMA specification which is the basis for this specification, in 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.

Resource: A defined Property Data with an associated inquiry for accessing the Property Data.

State: A collection of settings and/or dataset from a particular moment in time, which is available for Property Exchange to Get and Set. A Device may have one or more States available.

1.4 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 1 Words Relating to Specification Conformance

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

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

Table 2 Words Not Relating to Specification Conformance

Word	Reserved For	Notes
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 ResourceList integration

When a Responder receives an Inquiry: Get Property message with ResourceList, the Responder shall send a Reply to Get Inquiry message with a list of all Resources it supports so the Initiator understands its availability and settings. For more information see the Common Rules for MIDI-CI Property Exchange [MMA03].

A Responder that supports the Resources defined in this specification, StateList and State, shall declare such support in its reply to a ResourceList inquiry. See Sections 2.4 and 3.4.

2. Resource: StateList

2.1 Introduction

StateList is a List Resource used to get the list of currently stored States in a Device.

The resulting Property Data may be sent back to the Responder, or to another compatible Device, to set the Responder entirely to the transmitted State. For example, this mechanism may be used by an Initiator to configure a Device to a previous State upon the launch of a project.

A State with a "stateId" Property Value of "fullState" contains a combination of all other States in the Device. This is roughly a PE equivalent to a "MIDI Bulk Dump". Any other stateID is a subset of "fullState". To transfer the "fullState" might take a long time, whereas dealing with a small State change (for example, in a subset of the dataset) might be sufficient.

A Device that implements multiple Groups shall provide StateList Resources only on a single Group, preferably the first Group. Vendors should embed state information of other Groups into that single State Resource definition. Future versions of this specification might introduce per-group versions of StateList.

2.2 Initiator Requests Data from a Responder Using an Inquiry: Get Property Data Message.

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

Initiator Sends Inquiry: Get Property Data Message

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

2.3 Responder Sends Reply to Get Property Data Message

Responders that support the StateList Resource shall return an array of objects in the Property Data using a Reply to Get Property Data Message.

Each object in the array contains the following Properties:

Property Key	Property Value Type	Description
`title`	string, required	The name of the State being described
`stateId`	string (max 36 chars, "a-z", "0-9" or "_" characters only), required	This is the State ID of the particular Device State. It is used with the State Resource to Get or Set the State in "resId" Property.

`stateRev`	string	This is a Device-specific reference of the State in the MIDI Device. Devices may use this field to compare States to decide whether it is necessary to reload.
`timestamp`	integer	This is the Responder's concept of Unix time of the last update to this State. This is the number of seconds that have elapsed since the Unix epoch, that is the time 00:00:00 UTC on 1 January 1970, minus leap seconds. The Initiator is not required to synchronize it's clock to this property.
`description`	string, commonmark*	This is a text description that helps the user understand what this particular State does.
`size`	integer	This is the expected size of the State in bytes.

**See [MMA03] M2-103-UM Common Rules for MIDI-CI Property Exchange*

Responder Sends Reply to Get Property Data Message

Header Data	`{"status":200}`
Property Data	see below

```
[
   {
      "title": "User Programs",
      "stateId": "userPrograms",
      "stateRev": "ahd822nkdla",
      "timestamp": 1583330400,
      "description": "This is all of the User Programs",
      "size": 300
   },
   {
      "title": "Samples",
      "stateId": "samples",
      "stateRev": "dj8293n3289",
      "timestamp": 1579701600,
      "description": "A complete snapshot of all the Samples",
      "size": 2056789
   },
   {
      "title": "Buffer",
      "stateId": "buffer",
      "stateRev": "adoi234dvd",
      "timestamp": 1580652000,
      "description": "This is the current buffer RAM Memory",
      "size": 4456953
   }
]
```

It is expected that the initiator will be able to compare this data with locally stored States for this Device and compare the given stateRev for each State ID. If the stateRevs don't match then the user might be prompted to ask if they wish to replace the State with the locally stored, or if they wish to update the locally stored State with the one on the Responder.

2.4 ResourceList Integration for StateList

Example minimal entry in ResourceList:

Property Data

```
[
   {"resource":"StateList"}
]
```

Example full version with default settings:

Property Data

```
[
   {
      "resource": "StateList",
      "canGet": true,
      "canSet": "none",
      "canSubscribe": false,
      "canPaginate": false,
      "schema":{
         "type": "array",
         "title": "State List",
         "$ref": " http://schema.midi.org/property-exchange/M2-112-
S_v1-0_StateList.json"
      },
      "columns":[
         {"property": "title", "title": "Title"},
         {"property": "timestamp", "title": "UTC timestamp"},
         {"property": "description", "title": "Description"}
      ]
   }
]
```

3. Resource: State

3.1 Introduction

An Initiator uses the State Resource to Get or Set one of the States of the Responder, which it has discovered is available by the use of the StateList Resource. The resulting Property data may be sent back to the Responder, or another compatible Device, to set the Device's buffer to the transmitted State.

The "resId" Property used in the Header Data comes from the "stateId" Property returned in each entry from the StateList Property Data. For more information about "resId", see the Common Rules for MIDI-CI Property Exchange [MMA03].

3.1.1 Properties to Identify the State Property Data

In most cases, Property Data from a State Transaction is only useful for the Device that originally generated the Property Data (or another compatible Device of the exact same model and version). Therefore, it is strongly recommended that an Initiator store the Property Data in combination with the Device ID data obtained via the MIDI-CI Discovery Transaction or from a PE DeviceInfo Resource transaction. See the MIDI-CI Property Exchange Foundational Resources: DeviceInfo, ChannelList, JSONSchema [MMA04].

The Responder should include the current "stateRev" Property in the Header so that the Initiator can save the Property Value with stored State. This may be used to identify the State so that Initiator and Responder can determine if the State has changed since the last retrieval or backup and whether restoring a State might be unnecessary (if State has not changed).

The set of data to be stored should include Properties which identify the Device, Properties which identify the State, and the State Property Data as follows:

Device Properties:

1. Manufacturer SysEx ID ("manufacturerId" from the DeviceInfo Resource)
2. Device Family ("familyId" from the DeviceInfo Resource)
3. Device Family Model Number ("modelId" from the DeviceInfo Resource)
4. Software Revision Level ("versionId" from the DeviceInfo Resource)

State Properties:

5. State "timestamp"*
6. State "stateId" (as declared by the Initiator as "resId" in the Header Data of the inquiry)
7. State "stateRev"* (as declared by the Responder in the Header Data of the reply)

Property Data:

8. State Property Data (as declared by the Responder)

**Note: If the Responder cannot include the optional "timestamp" and/or "stateRev" Property, then the Initiator might store its current time and date as a tag for the data.*

3.2 Initiator Requests Data from a Responder Using an Inquiry: Get Property Data Message.

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

Initiator Sends Inquiry: Get Property Data Message

Header Data	`{"resource":"State","resId":"buffer","mutualEncoding":"Mcoded7"}`
Property Data	*none*

Responder Sends Reply to Get Property Data Message

Header Data	`{"status":200,"mutualEncoding":"Mcoded7","mediaType":"application/octet-stream","stateRev":"buui890adj","timestamp":1586786400}`
Property Data	*<Property Data encoded using Mcoded7>*

If the Responder has not declared the "size" Property for the Property Data, then the total data during transmission might be larger than then Initiator is able to use. In that case, the Initiator should send a Property Exchange Notify message containing a status code of 144 (Terminate Inquiry). This will inform the Responder to stop sending further data. See the Common Rules for MIDI-CI Property Exchange [MMA03], Section 10.2.

3.2.1 Format of Property Data

The format of the Property Data shall match all compression and encoding formats declared in the Header Data. If the Property Data is not already in a 7-bit format, then the Resource must make use of the "mutualEncoding" Header Property to ensure that the data will be compatible with the 7-bit format required by the rules of MIDI System Exclusive message payloads. See the Common Rules for MIDI-CI Property Exchange specification [MMA03] for details.

3.3 Send Data to a Device Using Inquiry: Set Property Data Message

An Initiator may send the Property Data to a compatible* Responder for the State Resource using an Inquiry: Set Property Data message.

**Note: The Initiator should get Device ID data to ensure that the target Responder is the same Device (or same model and version) as the original source of the data.*

Initiator Sends Inquiry: Set Property Data Message

Header Data	`{"resource":"State","resId":"buffer","mutualEncoding":"Mcoded7", "mediaType":"application/octet-stream"}`
Property Data	*`<Property Data encoded using Mcoded7>`*

Responder Sends Reply to Set Property Data Message

Header Data	`{"status":200,"stateRev":"fejwiofij2w","timestamp":1586786400}`
Property Data	*none*

The Responder should be able to determine that the State Property Data being sent by the Initiator is intended for that Responder. This may be achieved using information contained in the first bytes of the Property Data. A Device might also choose to include a device-specific checksum within the Property Data to confirm the quality of the data.

If a Responder detects that the data being transmitted by the Initiator is not correct during the transmission, then the Responder should send a Property Exchange Notify message containing a status code of 144 (Terminate Inquiry). This will inform the Initiator to stop sending further data. See the Common Rules for MIDI-CI Property Exchange [MMA03], Section 10.2.

3.4 "ResourceList" Integration for State

Example minimal entry in ResourceList:

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

Example full entry with default settings:

Property Data	`[`
	   `{`
	      `"resource": "State",`
	      `"canGet": true,`
	      `"canSet": "full",`
	      `"requireResId": true,`
	      `"canSubscribe": false,`
	      `"encodings":["Mcoded7"],`
	      `"mediaTypes":["application/octet-stream"],`
	      `"schema":{`
	         `"title": "State"`
	      `}`
	   `}`
	`]`

Appendix A: Common stateIds and Titles

Suggested States for Musical Instruments, for use as the "stateId" Property Value of objects in the StateList Property Data.

stateId	Title	Description
`fullState`	Full State	The entire collection of Device settings and data, including data from all other listed States.
`allNonSystem`	All Non System	The entire collection of Device settings and data other than the System Settings. (System Settings + All Non System= Full State)
`buffer`	Memory Buffer	The current working settings and data of the Device, sometimes called the edit buffer.
`system`	System Settings	The system, global, configuration, and utility settings of the Device.
`programs`	Programs	A complete set of all Program data in the Device.
`sequences`	Sequences	The sequences, (non-audio) loops, and/or MIDI files stored on the Device.
`arpeggios`	Arpeggiator Programs	Arpeggiator algorithms stored in the Device.
`samples`	Samples	The set of audio samples in the Device.
`effects`	Effect Programs	A complete set of all Effect Program data in the Device.

Devices are not limited to these States. The array of States, each with a "stateId" and "title", is flexible to allow any category to be declared as an object, defined by the Device manufacturer to suit the specific design of the Device.

For Example, some Devices might declare a set of custom States such as:

- Bank 1 and Bank 2
- Originals and Variations
- Multi-Samples and Loop Samples
- Full State, Buffer State, Bank 1, Bank 2, and Loop Samples

The descriptions of these suggested States are generalized. The actual content and details of each State is Device-specific. For example, "samples" in one Device might only be the raw audio data while in another Device "samples" might include meta data for the use of the audio data. It is up to each Device to determine which States are supported and the meaning of those States.

Revision History

Date	Version	Changes
Nov. 18, 2020	1.0	Initial Version

https://www.midi.org