CONTRIBUTION: Accessibility and the MIDI-CI Profile for Assistive Text

Music Accessibility Standard Special Interest Group
Mike Kent

2026-06-24, v3.0

This document proposes standardized mechanisms for MIDI Devices to communicate text to the accessibility interfaces which exist in common operating systems. Text data, with meta data, is used to enhance user interaction via external outputs such as speech, braille, or other assistive feedback mechanisms. The goal is to increase accessibility for blind and low-vision users, while being broadly useful to all MIDI users.

We propose that the MASSIG and MIDI Association should publish two documents:

1. Defines several mechanisms of a MIDI Text Receiver Utility Application in the OS to output assistive text from MIDI Devices
2. MIDI-CI Profile for Assistive Text defines how MIDI Devices integrate with the MIDI Text Receiver Utility Application.

This proposal focuses on enabling hardware devices. Software applications running on a computer have API access to assistive engines such as Text-to-Speech (TTS). Hardware devices do not have direct access to those same APIs. But nothing prevents MIDI software from using the mechanisms proposed here.

1 Overview Text Transfer and Text to Speech

MIDI devices connected to a computer can send and receive MIDI Data. Modern operating systems allow multiclient access to the connection. While the primary musical function might be provided by a DAW or other MIDI software application, a MIDI Text Receiver Utility application can also connect to the external device. The MIDI Text Receiver Utility application could receive text data from the external MIDI device and use the APIs in the OS to send the text to the Text-to-Speech engine and other assistive resources.

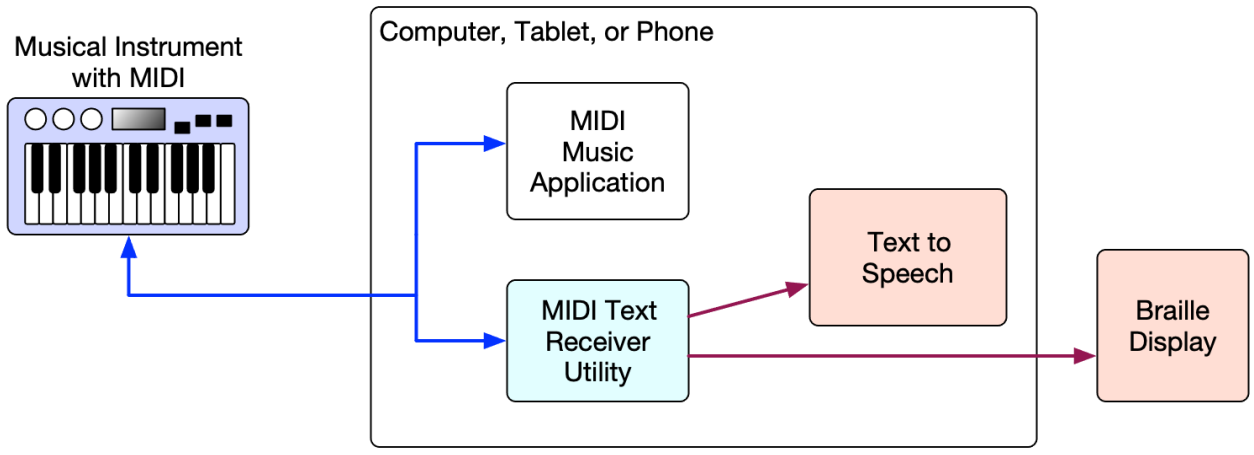

Figure 1 MIDI Device Connected to Computer

Text from the device might include:

- Knob/Button names
- Knob/Controller values
- Menu Item Readout

- Note Names
- Chord Names
- Help Articles

Or any other text which the manufacturer thinks will be useful.

2 MIDI Text Receiver Utility Application

The Utility Application should perform the following:

- Retrieve Text and Meta Data and forward it to the assistive engines which exist in the operating system.
- Provide a control panel for user configuration:

2.1 Retrieving Text and Meta Data

There are several possible sources of text and meta data. This proposal defines those as 4 Classes of functionality as outlined in Table 1 and further explained in Sections 3 to 6.

Table 1 Sources of Text and Meta Data

Class	Data	Utility Application Function	Implementation
1	Any MIDI Message. See Section 3.	The Utility Application provides some interpretation of any received MIDI messages such as notes, controllers, and program changes.	No change to any device. Works with every MIDI Device.
2	Independent Custom Data Set for Each Device. See Section 4.	The Utility Application reads a file which is specific to each manufacturer/model device. The file provides the data for the Utility Application to interpret MIDI messages in a more integrated manner. For example, the file could declare that CC#73 is Attack time ranging from 0 to 10 seconds. The host computer can navigate the menus of a hardware device.	Manufacturer or anyone else provides a data set file (probably JSON) which declares the names and values of properties which are attached to specific MIDI messages.
3	MIDI-CI Profile for Assistive Text. See Section 5.	The Utility Application receives data specifically designed for accessibility: text and meta data. We currently propose 3 mechanisms in the Profile: Text and Meta Data exist in the Device and are sent over the MIDI connection. The Primary Dictionary of common terms with Lookup IDs, in multiple languages. The Device sends an ID reference to point to an entry in the dictionary. There is an additional dictionary which is specific to a manufacturer/model device. The host computer can navigate the menus of a hardware device.	Device implements the Profile. The Utility Application implements the Profile.
4	Other MIDI-CI Profiles. See Section 6.	The Utility Application is aware of other Profiles. For example: The Utility Application can monitor the DAW Macro Control Profile report messages which declare destination parameter names and current values.	The Utility Application understands the Profile. See Section 6.

2.2 User Configuration Control Panel

The MIDI Text Receiver Utility application needs certain functions to ensure a holistic and comprehensive solution. Features which we might define in an Implementation Guide for the MIDI Text Receiver Utility Application might include (not a complete list yet):

- Select devices which are connected to the Utility Application
  - Discover Product Instance ID, to identify multiple units of the same model
- Customize the Utility Application experience (some need per device):
  - User selectable Prosody: Rate, pitch, volume, etc.
  - Set filters of the type of text to pass
  - Set filters to modify text, e.g. replace words
  - Allow the user to assign a custom name for each device
  - Prepended Device Chimes: to identify the device sending messages, when multiple devices are sending messages
  - Mute and Solo to select which devices are currently active
  - Select which Classes are currently active
  - Preferred Language
  - Add manufacturer/model specific dictionary
- Customize the experience by sending configuration commands to the Device
  - Verbosity
  - Preferred Language

Most of these features require the use of the MIDI-CI Profile for Assistive Text (see Section *5*).

The MIDI Association should discuss these ideas with the OS providers.

2.3 Connected to Several Assistive Interfaces

The MIDI Text Receiver Utility application could utilize multiple assistive outputs including Text to Speech, large text overlays, or an external braille display device.

The external MIDI device might have native support for multiple languages. But the MIDI Text Receiver Utility application could take advantage of language translation engines to support an even broader set of languages.

The MIDI Text Receiver Utility application might be a utility provided by the OS provider or could be provided by a third party (open source?). The MIDI Association should discuss these ideas with the OS providers.

3 Text Source Class 1: Interpret Simple MIDI Messages

The Utility Application could have a dictionary of text for common MIDI messages. The output could declare Note names (perhaps for music education), controller values, program change values (perhaps including names if GM2), pitch bend values, Start, Stop, etc. This is not a sophisticated feature, but implementation is simple and this works with all MIDI devices made since 1983. Devices do not need any update.

4 Text Source Class 2: Custom Data Set per Device

Manufacturers (or other people) provide a file which declares specific interpretations of MIDI messages which the Device will send. For example, the file could declare that CC#73 is Attack time ranging from 0 to 10 seconds. This mechanism is well suited to devices which have a knob per function design, such as knobby analog synthesizers. Devices do not need any update, but a custom data set file is required.

> *Note: This mechanism is not suitable for typical MIDI Controllers because the destination parameter is unknown and changes often. Class 3 and Class 4 are better suited to MIDI Controllers.*

5 Text Source Class 3: MIDI-CI Profile for Assistive Text

The mechanisms in this proposal should be implemented in a MIDI-CI Profile specification. A Profile is a defined set of rules for how MIDI devices implement a chosen set of MIDI messages to achieve a particular purpose or to suit a particular application. Devices which implement a Profile have a high level of interoperability with each other. To use this Profile, the steps might be:

1. The MIDI Text Receiver Utility application uses the MIDI-CI profile Inquiry to determine if the connected device supports the MIDI-CI Profile for Assistive Text. If the device replies that it supports the Profile, then the Profile is enabled on the Device.
2. The MIDI Text Receiver Utility application uses a MIDI-CI Profile Details Inquiry to discover what type messages and other options that the Device supports. Then the MIDI Text Receiver Utility application may optionally configure the device. For example, the MIDI Text Receiver Utility application may discover that the MIDI Device has text in four languages and tell the device which one is preferred.
3. MIDI Device starts sending text for related events.

5.1 MIDI Messages with Text

The MIDI 2.0 Protocol provides several message types which are suitable for sending text data. We should examine the pros and cons of each type to determine which types might be the most suitable. We might use one message format for all text. Or we might use multiple types, for example one format for short data of a word or phrase and a different format for a long set of text such as a tutorial article of several sentences or paragraphs.

5.2 MIDI Messages with Short Text

I propose several methods for sending short text including individual words, property values, phrases, or a sentence. Following are several proposed mechanisms, as described in further sections:

- Flex Data message for text sent from the device
- Flex Data message with an ID used for lookup in a dictionary used by the Utility Software
- SysEx message for text sent from the device (works on MIDI 1.0 devices)

5.3 Flex Data Message: Text From the Device

Flex Data messages can be used for a text payload with a maximum size 360 bytes (but more friendly for less than the maximum). This makes it suitable for single words, phrases of several words, or a single sentence.

The message can contain several fields of meta data.

The standard encoding of text in Flex Data messages is UTF-8 format. However, it will be possible to define the use of UTF-16, assuming that we will find that is preferable (most TTS systems use UTF-16).

Figure 2 shows an example, with several fields of meta data followed by the text data. This is just an example, not a specific proposal for which fields we should include.

Text From Device: Header UMP

mt=d	group	f=0/1	addrs	channel	status bank = 0x02	status = 0x20
priority		utterance type			source type	text data / reserved
text data / reserved		text data / reserved			text data / reserved	text data / reserved
text data / reserved		text data / reserved			text data / reserved	text data / reserved

Text From Device: Further Multiple UMPs as Necessary to Complete the Text (max = 31 Further UMPS)

mt=d	group	f=2/3	addrs	channel	status bank = 0x02	status = 0x20
text data / reserved		text data / reserved			text data / reserved	text data / reserved
text data / reserved		text data / reserved			text data / reserved	text data / reserved
text data / reserved		text data / reserved			text data / reserved	text data / reserved

Figure 2 Flex Data Message: Assistive Text

Not shown here is the possibility to define a delimiter at the end of the text; then optional data after the delimiter to further define the pronunciation.

5.4 Flex Data Message: ID of Text in a Dictionary

The MIDI Text Receiver Utility application could have a Dictionary of very common terms. Flex Data messages can be used to select text from the dictionary using a reference ID value. This might typically require fewer resources in the device.

Figure 3 shows an example. The final design should probably have payload which includes the same meta data as the Flex Data Message: Text From the Device we define (see Section 5.1).

Text ID Reference to Dictionary

mt=d	group	f=0	addrs	channel	status bank = 0x02	status = 0x21
priority		utterance type			source type	reserved
dictionary id					dictionary language	
id of text in dictionary						

Figure 3 Flex Data Message: Assistive Text ID Reference

This supports many dictionaries. Some dictionaries (in particular, a primary dictionary with a core set of terms we define as the very common and standard terms always available) might include multiple languages.

We probably need to define additional mechanisms to make dictionaries most useful:

- A way for a Device to get a list of dictionaries that the MIDI Text Utility Application has available.
- A way for the MIDI Text Receiver Utility application to get custom dictionaries from connected devices.

5.5 System Exclusive Text Payloads

System Exclusive is very flexible to transport any data, small or large. System Exclusive gives one notable advantage: it can be used on MIDI 1.0 transports.

However, there are drawbacks to using System Exclusive. The data format is limited to 7 bits of data per byte. To support UTF-16, we will need an encoding mechanism (see “Mcoded7” in the UMP and MIDI 2.0 Protocol specification).

The MIDI-CI Profile mechanism defines a method for transferring data which directly related to the Profile.

Table 2 shows an example, with a payload which includes several fields of meta data followed by the text data. This is just an example, not a specific proposal for which fields we would include. The final design should have payload which is the same as the data in the Flex Data message we define (see Section 5.1).

Table 2 Profile Specific Data Message for Assistive Text

<table>
<tr><th>Value</th><th>Parameter</th></tr>
<tr><td>F0</td><td>System Exclusive Start</td></tr>
<tr><td>7E</td><td>Universal System Exclusive</td></tr>
<tr><td>1 byte</td><td>Device ID: Source or Destination (depending on type of message):
00–0F: To/from MIDI Channels 1-16
7E: To/from Group
7F: To/from Function Block</td></tr>
<tr><td>0D</td><td>Universal System Exclusive Sub-ID#1: MIDI-CI</td></tr>
<tr><td>2F</td><td>Universal System Exclusive Sub-ID#2: Profile Specific Data</td></tr>
<tr><td>1 byte</td><td>MIDI-CI Message Version/Format</td></tr>
<tr><td>4 bytes</td><td>Source MUID (LSB first)</td></tr>
<tr><td>4 bytes</td><td>Destination MUID (LSB first)*</td></tr>
<tr><td>5 bytes</td><td>Profile ID</td></tr>
<tr><td>4 bytes</td><td>Length of Following Profile Specific Data (LSB first)</td></tr>
<tr><td>nn bytes</td><td>Profile Specific Data
<table>
<tr><td>1 byte</td><td>Priority</td></tr>
<tr><td>1 byte</td><td>Utterance Type</td></tr>
<tr><td>1 byte</td><td>Source Type</td></tr>
<tr><td>nn-3 Bytes</td><td>Text Data</td></tr>
</table></td></tr>
<tr><td>F7</td><td>End Universal System Exclusive</td></tr>
</table>

There is a UMP / MIDI 2.0 equivalent of this System Exclusive message. However, if a Device supports the UMP Format, then it is better to use Flex Data messages (see Section 5.2 and 5.3) in most cases.

5.6 MIDI Messages with Long Text

We could define mechanisms for long text or complex data. But the priority for this is probably lower than for short text proposed above. This might even be a separate Profile.

5.7 System Exclusive 8

System Exclusive 8 is very similar to legacy System Exclusive (above) but data is more friendly, using 8bits per byte. Data is sent in 128-bit UMP packets, so the number of MIDI messages used to transfer data are greatly reduced as compared to legacy System Exclusive. System Exclusive 8 messages require the use of MIDI 2.0 systems. System Exclusive 8 could be used to transfer one or more pages of text. This might be used for data such as a help article provided in context of the current activities or features being used on a device.

5.8 Utility Application Configures the Device

MIDI Text Receiver Utility application might send commands to the external hardware device. In some cases, the hardware device should (optionally) acknowledge those commands.

5.8.1 Language Configuration Messages

The MIDI Text Receiver Utility application could send a configuration message to request a specific language from a device. This might be a user setting in the MIDI Text Receiver Utility or might be set by the default language of the OS. I propose that we use the language tags defined in IETF BCP 47. The core language is coded in 2 bytes. Following bytes are optional and further define details including script, region, and custom data.

Flex Data Language Configuration Request

mt=d	group	f=0/1	addrs	channel	status bank = 0x02	status = 0x22
language					language extended / reserved	language extended / reserved
language extended / reserved		language extended / reserved			language extended / reserved	language extended / reserved
language extended / reserved		language extended / reserved			language extended / reserved	language extended / reserved

Further Multiple UMPs as Necessary to Complete the Language Configuration

mt=d	group	f=2/3	addrs	channel	status bank = 0x02	status = 0x22
language extended / reserved		language extended / reserved			language extended / reserved	language extended / reserved
language extended / reserved		language extended / reserved			language extended / reserved	language extended / reserved
language extended / reserved		language extended / reserved			language extended / reserved	language extended / reserved

Figure 4 Flex Data Message: Language Configuration Request

If a Device receives a Language Configuration Request, then it should reply to acknowledge the change of language, or to report the language chosen by the device if the request cannot be honored. The Device may also send a message to inform the MIDI Text Receiver Utility application which language it will be sending if the user makes a selection on the Device.

Flex Data Language Configuration Report

mt=d	group	f=0/1	addrs	channel	status bank = 0x02	status = 0x23
language					language extended / reserved	language extended / reserved
language extended / reserved		language extended / reserved			language extended / reserved	language extended / reserved
language extended / reserved		language extended / reserved			language extended / reserved	language extended / reserved

Further Multiple UMPs as Necessary to Complete the Language Configuration

mt=d	group	f=2/3	addrs	channel	status bank = 0x02	status = 0x23
language extended / reserved		language extended / reserved			language extended / reserved	language extended / reserved
language extended / reserved		language extended / reserved			language extended / reserved	language extended / reserved
language extended / reserved		language extended / reserved			language extended / reserved	language extended / reserved

Figure 5 Flex Data Message: Language Configuration Report

5.8.2 Verbosity Level Configuration

The MIDI Text Receiver Utility application could send a message to request a level of verbosity from a device.

Flex Data Verbosity Configuration Request

Figure 6 Flex Data Message: Verbosity Level Configuration Request

If a Device receives a Verbosity Level Configuration Request, then it should reply to acknowledge the change of verbosity, or to report the verbosity level chosen by the device if the request cannot be honored. The Device may also send a message to inform the MIDI Text Receiver Utility application of a verbosity level setting if the user makes a selection on the Device.

Flex Data Verbosity Configuration Report

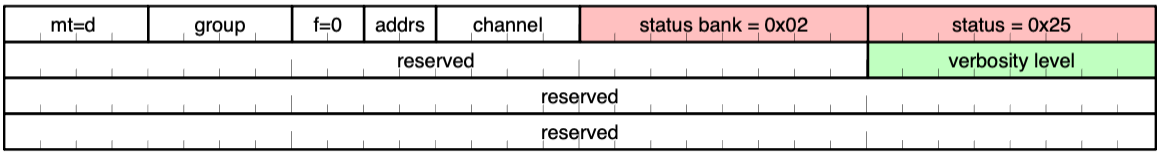

Figure 7 Flex Data Message: Verbosity Level Configuration Report

6 Text Source Class 4: Use Other MIDI-CI Profiles

MIDI-CI Profiles define which MIDI messages are used for specific tasks. If the MIDI Text Receiver Utility Application understands the Profile, it can provide very specific information about the use of messages exchanged in the Profile.

> For example, if the Utility Application is aware of the Piano Profile, it could interpret a MIDI 2.0 RC 0x41 0x07 with a value of 0x80000000 and create the text " Piano Lid Half Stick".

It may not be feasible for the Utility Application to understand all messages of all Profiles (especially for version 1.0).

However, we recommend that the Utility Application should focus on implementing the DAW Macro Control Profile. In this Profile, an External MIDI Controller sends commands to the DAW and the DAW responds. The Utility Application could voice the DAW response.

> For example, consider a MIDI controller with 8 knobs connected to a DAW with a Macro function. The knobs send a fixed set of controllers, and the DAW routes them dynamically to various functions. The destination parameter and meaning of the value change regularly, often based on the current focus. The DAW always determines and knows the destination.
>
> When the DAW Macro Control Profile is active, the DAW responds to the MIDI Controller by reporting parameter names and values. When the user turns a knob on the MIDI Controller, the Utility Application could express the destination parameter name ("Filter Cutoff") which it got as text from the DAW. It could express the value it got from the DAW (value = 0x35C28F5C, best expressed as "21%") or the text value it got from the DAW ("3012Hz").

The Utility Application should also understand the Default Control Change Mapping Profile.

> For example, CC#10 is always Pan, so the user should hear informative text such as "Pan 30% Left".

7 From the Prior Version of this Proposal – Outdated?

The Profile should focus on delivering text. We want to keep it simple for hardware developers to implement.

However, assistive mechanisms have APIs for controls and for meta data such as language, emphasis, timing, and segmentation. We need to determine which of these properties should be in the MIDI message protocol, which are properties that should be handled by the MIDI Text Receiver Utility application, and which are out of scope.

Properties to study might include (but are not limited to):

- Language Tag (probably IETF BCP 47)
- Utterance type (word, phrase, sentence, paragraphs)
- Device Type: Menu item, knob/button name, knob/button value, note name, chord name, etc.
- Prosody: Rate, pitch, volume Rate
- Start, pause, finish controls
- Word boundary markers
- Queuing: Speak immediately or enqueue
- Markup

Perhaps we use the following as a general guideline: Is there anything which the MIDI message can include to help the MIDI Text Receiver Utility application make the best use of the OS APIs? For those properties which are common to most/all systems, we might consider including the relevant data in the MIDI message stream. Those properties which are very specific to the API of one operating system should be in the domain of the MIDI Text Receiver Utility application.

7.1 Property Exchange

Property Exchange uses System Exclusive messages to inquire, get, and set properties in a device. Property Exchange is designed to transfer structured data in JSON format. It can be used for a small data set, but has a defined chunking mechanism for large data sets. Property Exchange can also be used to retrieve from a device any complex document with text or other media, such as a dictionary, an owners' manual or help video file.

Property Exchange requires a more advance set of bidirectional transactions that any of the mechanisms proposed above.

http://www.amei.or.jp

https://www.midi.org