MIDI PLUGIN

View online: https://www.construct.net/en/make-games/manuals/construct-3/plugin-reference/midi

The **MIDI plugin** allows communicating with connected MIDI (Musical Instrument Digital Interface) devices via the Web MIDI API. Typically MIDI devices are digital musical instruments such as a MIDI piano keyboard or synthesizer. The MIDI plugin can both receive MIDI input from devices, such as detecting when notes are pressed, as well as send MIDI output to devices, such as to create an arpeggiator.

In order to use a MIDI device it will need to be connected to the device that the Construct project is running on. Some MIDI devices may require a special 5-pin MIDI cable, but others may be able to use another standard such as USB.

You can find two examples of using the MIDI plugin in the MIDI input and MIDI output examples.

Familiarity with the MIDI protocol may be useful when using MIDI. You can search the web for the full specification, but there's also a concise unofficial third-party reference here.

As of February 2025, Safari does not support the Web MIDI API and so the MIDI plugin will not work in web exports loaded in Safari, or in macOS WKWebView exports. Use the Is supported condition to check if MIDI is supported on the current browser/platform.

MIDI access

For privacy and security reasons, browsers do not provide access to MIDI devices by default. Therefore the first thing your project must do before using MIDI is use the *Request MIDI access* action. This may show a permission prompt to the user. This action may also only be allowed in a user input trigger, such as in a button click or touch event, although some browsers may allow it to be used on startup (in *On start of layout*).

The user must approve any permission prompt - if they do so, *On request access success* is triggered, and MIDI devices may then be used. If the user declines the permission prompt *On request access failed* will be triggered and access to MIDI will be blocked. In this case you may wish to show a message to the user.

There are two options when requesting MIDI access:

- **System exclusive:** when enabled, this allows for sending and receiving system exclusive (sysex) MIDI messages. These are usually device or vendor specific. As these messages have potentially unlimited capability, such as wiping data stored on the device or overwriting its firmware, there may be stricter security requirements for using this mode; therefore it is recommended to leave it disabled if you don't need it.

- **Software:** when enabled then virtual or software MIDI devices running on the same system as the Construct project may be listed as available. If you are only interested in using separate physical MIDI hardware, you should leave this disabled.

MIDI devices

By default, after successfully gaining MIDI access, no MIDI messages will be sent or received. To use a MIDI device, it must first be *opened*. This means the device becomes active and messages may be sent or received. If the device is no longer needed it may then be *closed* to return it to an inactive state that will not send or receive MIDI messages.

There are two types of MIDI devices: input, from which messages are received, and output, to which messages may be sent. If a single piece of MIDI hardware supports both input and output, it will appear as two devices - both an input device and an output device. Sometimes this will also require two cables, one for input and one for output. Be sure to make sure the appropriate MIDI sockets are used - for example to receive MIDI input from a device, you may need to connect a cable from the device's MIDI output socket to the MIDI input socket of the device the Construct project is running on.

The MIDI plugin supports input and output to multiple devices simultaneously. To allow for this, devices are identified by an ID string provided by the browser or system. Usually the ID string can be left empty to refer to either the first available device, or the last opened device, which is a convenient way to use a single device.

MIDI notes

In the MIDI specification, notes are identified by a number from 0-127, with number 60 referring to middle C. With Construct's MIDI plugin, for convenience notes may also be referred to by a string of the note name. For example number 60 (middle C) may be referred to by the string "C4", meaning note C in octave 4.

Note names may also use a sharp ("#") or flat ("b") to adjust the note by a semitone in either direction, for example "D#4" or "Db4". This means both "C#4" and "Db4" refer to the same note. Construct also treats "Cb" as equivalent to "B", and "B#" as equivalent to "C".

The *NoteNumberToName* and *NoteNameToNumber* expressions allow converting between note number and note name representations. Note that when converting from a note number to a note name, Construct will always use a sharp notation. For example note 61 is both "C#4" and "Db4", and Construct will choose the name "C#4" in this case.

MIDI conditions

Has access

True if MIDI access is available, i.e. requesting MIDI access has completed successfully.

Is supported

True if MIDI access is supported on this browser/platform. This depends on support for the Web MIDI API.

On devices state changed

Triggered when any MIDI devices are connected, disconnected, or changed. If your project displays a list of available MIDI devices, it should update the list in this trigger.

> *This may trigger multiple times in quick succession - for example if a single MIDI hardware device that provides both an input and an output is removed, then this trigger may fire twice as both the input and output become unavailable.*

On request access success

On request access failed

Triggered after the *Request MIDI access* action, depending on the outcome of the request.

On any device opened

On device opened

Triggered after the *Open device* action when the device has been opened and can now send or receive MIDI messages. With the *any* variant, the relevant MIDI device can be identified by expressions like *CurrentDeviceID*.

On any device closed

On device closed

Triggered after the *Close device* action when the device has been closed and so will no longer send or receive MIDI messages. With the *any* variant, the relevant MIDI device can be identified by expressions like *CurrentDeviceID*.

On any device error

On device error

Triggered if an error occurs when accessing a MIDI device. With the *any* variant, the relevant MIDI device can be identified by expressions like *CurrentDeviceID*.

Compare parameter

Compare an aspect of a MIDI message such as the note number or velocity.

On any binary message

Triggered when any MIDI message is received from any device. The raw binary data of the MIDI message is stored in a specified Binary Data object. The associated MIDI device can be identified by expressions like *CurrentDeviceID*. Using the binary data will require knowledge of the MIDI specification - for example a three-byte message in hexadecimal notation 0x90 0x3C 0x40 refers to Note On, MIDI channel 0, note 60 (middle C), velocity 64.

On any message

Triggered when a MIDI message of a specific type, e.g. Note On, is received from any MIDI device. The associated MIDI device can be identified by expressions like *CurrentDeviceID*. Details about the message can be compared with *Compare parameter* or retrieved via expressions.

On message

Triggered when a MIDI message of a specific type, e.g. Note On, is received from a specific MIDI device given by its ID. The device ID may be left as an empty string to refer to the last opened input device. Details about the message can be compared with *Compare parameter* or retrieved via expressions.

MIDI actions

Request MIDI access

Request to access MIDI devices. This must be done before any MIDI devices can be used, and may show a permission prompt to the user. For more details see the section on *MIDI access* above.

Open device

Open a MIDI device specified by its device ID and type (input/output). The device ID may be left as an empty string to open the first available device; however note if multiple devices are available, this may not select the device you really want, so generally you should provide a way for the user to pick the device (see the MIDI input/output examples in Construct). Once a device is opened, it may start sending or receiving messages (depending on whether it is an input or output device).

Close device

Close a MIDI device specified by its device ID and type (input/output). The device ID may be left as an empty string to close the last opened device (useful when only one device is opened at a time). Once a device is closed, it will no longer send or receive messages.

Schedule next message

This action will cause the next *Send* action to actually send the message after a given delay in seconds. This can be used to schedule messages to be sent with higher accuracy than ticks in Construct, which typically run approximately every 16ms (for a 60 Hz display).

Send binary message

Send raw binary data as a MIDI message to an output device ID. The ID may be left as an empty string to send a message to the last opened MIDI output device. The message contents is taken from a chosen Binary Data object. Using the binary data will require knowledge of the MIDI specification - for example a three-byte message in hexadecimal notation 0x90 0x3C 0x40 refers to Note On, MIDI channel 0, note 60 (middle C), velocity 64.

Send control change message

Send a MIDI Control Change message to an output device ID. The ID may be left as an empty string to send a message to the last opened MIDI output device. Refer to the MIDI specification for possible controller numbers; a common one to use is controller number 1 to refer to the modulation wheel (coarse). The controller value is a number in the range 0-127.

Send message

Send a Program Change, Channel Pressure or Pitch Wheel message to an output device ID. The ID may be left as an empty string to send a message to the last opened MIDI output device. The value is a number in the range 0-127 for Program Change or Channel Pressure messages, or 0-16383 for Pitch Wheel messages.

Send note message

Send a Note Off, Note On or AfterTouch message to an output device ID. The ID may be left as an empty string to send a message to the last opened MIDI output device. The note may be specified by either its MIDI note number (0-127, e.g. 60 for middle C) or a note name e.g. "C4". The note velocity is given as a number 0-127.

> *Some devices will treat a Note On message with velocity 0 as Note Off.*

Set MIDI channel

Set the MIDI channel of outgoing messages, from 0-15. The default is channel 0. Some devices will be configured to only receive messages on a certain channel and will have a setting to select the MIDI channel number of the device; note that many consumer devices will use 1-based indexing for the MIDI channel (so the first MIDI channel is displayed as channel 1, rather than channel 0).

MIDI expressions

InputCount

InputIDAt(index)

InputManufacturerAt(index)

InputNameAt(index)

Retrieve the list of available input devices. These expressions provide the number of input devices, and the device ID, manufacturer, and name of the device at a given zero-based index. Note this list first becomes available in *On request access success*, and may change in *On devices state changed*.

OutputCount

OutputIDAt(index)

OutputManufacturerAt(index)

OutputNameAt(index)

Retrieve the list of available input devices. These expressions provide the number of input devices, and the device ID, manufacturer, and name of the device at a given zero-based index. Note this list first becomes available in *On request access success*, and may change in *On devices state changed*.

CurrentDeviceID

CurrentDeviceManfucaturer

CurrentDeviceName

In a trigger, such as *On any message*, these expressions contain information about the device that sent the message.

Channel

When a MIDI message is received, this returns the MIDI channel of the message from 0-15.

ControllerNumber

ControllerValue

When a Control Change message is received, these return the controller number (0-127) and controller value (0-127) specified in the message.

NoteName

NoteNumber

When a note message is recieved (Note Off, Note On or After Touch), these return the associated note name as a string (e.g. "C4") or MIDI note number (0-127, e.g. 60 for middle C).

PitchWheelValue

When a Pitch Wheel message is received, this returns pitch wheel position in the range 0-16383.

Pressure

When an After Touch or Channel Pressure message is received, this returns the pressure value (0-127).

ProgramNumber

When a Program Change message is received, this returns the program number (0-127).

Velocity

When a Note On or Note Off message is received, this returns the velocity in the range 0-127. If the device does not support detecting note velocity, a default value such as 64 may be sent instead. Some devices send a Note On message with velocity 0 to indicate Note Off; in this case the velocity will be returned as 64 as a default value given the velocity is not actually specified in that case.

NoteNameToNumber(NoteName)

NoteNumberToName(NoteNumber)

Convert between a note name (e.g. "C4" for middle C) and a MIDI note number in the range 0-127 (e.g. 60 for middle C). See the section on *MIDI notes* above for more information.

> *NoteNumberToName will always use the sharp variant, e.g. note number 61 will always return "C#4" and never "Db4".*