SIBELIUS®

THE MUSIC NOTATION SOFTWARE

MIDI messages

March 2001 – Edition 1.4

MIDI messages

(For MIDI specialists only!)

MIDI is that most rare of beasts, a standard set by a number of different manufacturers that is universally implemented and supported. This sounds too good to be true, and it is, because in order to understand exactly how MIDI works, you need to be able to speak Martian.

MIDI instruments (such as your computer's sound card or external sound module) send and receive MIDI messages, which consist of a *status byte* and one or two *data bytes*. MIDI bytes can have a decimal value of 0-127, which is their most human-readable form (they can also be addressed in terms of seven decimal bits – e.g. 1111111 – or hexadecimal up to 7F, but unless you're a computer, you won't want to think about this for too long).

In order to be device-independent, numbers in MIDI messages (including program numbers) always count from 0, even if your MIDI device's manual counts from 1.

MIDI messages are classified either as *channel messages*, which affect a single channel (in Sibelius, this translates to the staff to which they are attached), and *system messages*, which affect all channels (in Sibelius, all staves).

Channel messages carry the majority of the musical data (e.g. which notes to play, how long they should last, which sound to use), while system messages are used for other arcane things like synchronization with other MIDI devices.

Sibelius supports most MIDI messages (e.g. pitch bend, SYSEX), and can also send just raw data.

Entering MIDI messages in Sibelius

In general, you'll only need to enter channel messages manually in Sibelius in very specific circumstances, because Sibelius reads almost all markings in your score and automatically turns them into appropriate MIDI messages when playing back. You can also define words and phrases and what their effect on playback is using the playback dictionary.

However, you can also tell Sibelius to send any MIDI message you like at any point, by typing it in as text using the simple MIDI message language described below. These MIDI messages can be appended to ordinary text and can be hidden, so if you write **mute ~P59** in a trumpet part, Sibelius will send program change 59 to produce a muted trumpet sound at the exact point where 'mute' appears on the print-out. The **~P59** can be hidden. (Clever, eh?)

If the trumpet mutes often, you can of course copy **mute ~P59** using ⌥-click *or* **Alt**+click to save you retyping, or you can add it to the word menu obtained when you **Control**-click (Mac) *or* right-click (Windows) (🕮 **Word menus and special characters**), and assign it a keyboard shortcut at the same time.

When you import a MIDI file, you can choose to have MIDI messages in the file written into your score, in which case any control changes etc. will appear as if you'd typed them in yourself.

Should I use the dictionary or MIDI messages?

Many situations are more conveniently dealt with using the playback dictionary. If you want to send a MIDI message which is always associated with a particular word in your score, then just define it as a new word in the dictionary.

For instance, if trumpet is the only instrument in your score which is ever told to 'mute', then just define 'mute' to send program change 59 in the dictionary, and you won't have to keep typing **~P59** all over the place.

However, if your score uses muted strings as well as trumpet, then using the playback dictionary would make the strings also produce a muted trumpet sound wherever they were told to 'mute'.

In this case, you should use MIDI messages, so you can send a muted trumpet program change wherever the trumpet mutes, and turn the volume down (say) wherever the strings mute.

Syntax

You can type MIDI messages into your score using any staff text style – typically Technique or Expression text. MIDI messages can be written on their own, or put at the end of any other text (such as 'mute').

Messages take the form: **~** followed by a single command letter, followed by one or more numbers, which are usually separated by commas.

E.g. **~C64,127**

(**~** is informally called a 'tilde' or 'swung dash', but the technical term is 'twiddle'.)

Numbers can be written in decimal (e.g. **0-127**), binary preceded by **b** (e.g. **b01111010**), or hexadecimal preceded by **h** (e.g. **hA8**).

Note that:

- MIDI messages are case sensitive (i.e. you must type capitals or small letters as indicated), except for hex digits – so **~P123** is correct but **~p123** won't work
- You can write multiple messages in the same piece of text, separated by a space or **Return** (on the main keyboard), and with just one twiddle at the start, e.g.: **~P43 A65 C64,127**
- If you like you can also put spaces or **Return**s around commas and numbers.

Channel messages

Channel messages are split into two types: *channel voice messages*, which carry the musical data; and *channel mode messages*, which affect how the MIDI device responds to the musical data.

Let's examine each of the channel messages in turn:

NoteOn/NoteOff

These messages control which pitch is played, how loud the note is, and how long it lasts for. These cannot currently be entered manually in Sibelius. They consist of two data bytes (MIDI key number – e.g. 60 is middle C – and velocity), although the second byte is usually ignored in NoteOff messages.

Program and bank change

A program change controls which sound is used to play a particular note. Sibelius automatically sends the correct program change when it starts playing, so that a violin staff plays with a violin sound. However, if you want to change the sound a staff plays with midway through your score, you can use a program change message.

The syntax of a program change is **~P** *program* e.g. **P76**

Sibelius also allows you to change the bank and program in the same MIDI message. If your playback device only supports GM sounds, you'll never need to use a bank and program change together, but if it has a wider selection of sounds, you may want to use a sound from a different bank.

The syntax of a bank and program change is: **~P** *bank*/*program* e.g. **P2/76**

Note that program numbers here always start from 0, and bank numbers are always represented as (MSB x 128)+LSB. This is not necessarily how the values appear in the **Sounds** dialog box (**Play/Flexi-time** menu).

Pitch bend

Pitch bend normally allows you to alter the pitch of a note by up to two semitones up or down (although there are a couple of ways to increase its range – see below).

The syntax of pitch bend is **~B0,***bend-by*, e.g. **~B0,64** means normal pitch, **~B0,0** means bend two semitones downwards, and **~B0,127** means bend two semitones upwards. (In other words, values lower than **64** flatten the note, and values higher than **64** sharpen it.)

If you want finer control over the pitch bend, you can change the initial byte, also in the range 0-127, which gives very small deviations in temperament, so e.g. **~B127,64** will sharpen the written note by a small amount.

You can use the pitch bend control to create a portamento or glissando effect by inserting a number of MIDI messages one after another.

You can also use this control to make a note play back sharp or flat without inserting an accidental, e.g. if you want to make *ficta* – editorial accidentals above the staff – play back, you can insert the accidental from the **Symbols** dialog box, and then use a MIDI message of e.g. **~B0,96** to play the note a semitone sharp. Don't forget to use **~B0,64** to return the channel to its normal tuning on the next note!

To create a pitch bend effect over an interval wider than two tones, you can either use the portamento control change (see **Control changes** below) to make a pitch bend, or use the following method:

- First, set up the range over which the pitch bend can operate. Insert the MIDI messages **~C101,0 ~C100,0 ~C6,***semitones* in your score, where *semitones* is the total range of the pitch bend in semitones, from 0-12. For example, to set up pitch bend with a maximum range of an octave, use **~C6,12**.
  (It's best to put these messages at the start of your score.)
- When you want to add a pitch bend to your score, insert a **~B0,***bend-by* command as usual, except that now you must divide the *bend-by* parameter into the number of semitones set up with your **~C6** command, e.g. if you enter **~C6,12**, each semitone adds or subtracts 5.3 (64 divided by 12) to *bend-by*. So to bend upwards by four semitones, you would enter **~B0,85.2**.

Note that this method requires that your MIDI device supports standard 'Registered Parameters', which is common but not universal.

Aftertouch

Aftertouch refers to the amount of pressure used when e.g. a key on a MIDI keyboard is pressed. This information can be used to control some aspects of the sound produced by the synthesizer, e.g. vibrato on a violin sound. The precise effect of this controller is dependent on the sophistication of the MIDI device used.

Aftertouch can be applied either on a particular note (*polyphonic aftertouch*), or on a channel as a whole (*channel aftertouch)*. Polyphonic aftertouch is not as widely implemented in MIDI devices as channel aftertouch.

The syntax is as follows:

- Channel aftertouch: ~A *pressure* e.g. **A64**
- Polyphonic aftertouch: ~a *pitch*,*pressure* e.g. **a60,64**

Control changes

Control changes are used to control a wide variety of functions in a synthesizer. Although the function of each control change is clearly defined, not all MIDI devices support every control change. These are split up into groups, including:

- Control changes 0-31: data from switches, modulation wheels, faders and pedals on the MIDI device (including modulation, volume, expression, etc.)
- Control changes 32-63: optionally send the LSB for control changes 0-31 respectively
- Control changes 64-67: switched functions (i.e. either on or off) such as portamento, sustain pedal, damper (soft) pedal, etc.
- Control changes 91-95: depth or level of special effects such as reverb, chorus, etc.
- Control changes 96-101: used in conjunction with control changes 6 and 38 (Data Entry), these can be used to edit sound patches
- Control changes 121-127: channel mode messages (see **Channel mode messages** below).

The syntax for control changes is ~C*byte1*,*byte2*, where *byte1* is the number of the control change (from 0-127) and *byte2* is the control value (also from 0-127).

For full details of the control changes supported by your MIDI device, consult the manufacturer's manual.

The most commonly used control changes are as follows:

Modulation

Control change 1 controls the vibrato 'wobble' generated by a modulation wheel. For lots of wobble, use **~C1,127**; for no wobble, use **~C1,0**.

Breath

Control change 2 is only recognized by certain MIDI equipment, such as wind synthesizers, and corresponds to the 'breath pressure' used to play a note.

Portamento

Portamento is actually controlled by two control changes: **~C5**,*0-127* controls the length of time taken to perform the portamento (0 is fastest, 127 is slowest), and **~C84**,*0-127* determines the distance of the portamento (values below 60 start below the note, values above 60 start above the note).

So you must first 'set up' the portamento effect with a **~C5** message, and then attach the **~C84** message to the note on which the portamento occurs.

Note that this control change is not supported by all MIDI equipment.

Volume

Controller 7 determines the volume of a given note, e.g. **~C7,127** is the loudest and **~C7,0** is the softest.

You should only need to use this MIDI message if you want to achieve a change of dynamic over the course of a held note – the **Cresc./Dim. Playback** plug-in enters these messages for you.

Pan

Controller 10 determines the pan position of a particular channel, e.g. **~C10,0** is absolute left, **~C10,64** is center, and **~C10,127** is absolute right.

You don't need to use this MIDI message unless you need to change the pan position of an instrument during playback – you can usually change this setting in the **Sounds** dialog box (type **I**).

Expression

Controller 11 takes a fraction of the channel volume specified by controller 7, so **~C11,127** uses 100% of the channel volume, **~C11,64** uses 50% of the channel volume, and so on. Functionally **~C7** is intended to act something like a volume knob and **~C11** something like a pedal for tweaking the 'main' volume.

Sustain pedal

Sibelius automatically inserts MIDI messages for the sustain pedal if you use the *Ped.* lines from the **Lines** dialog box (📖 **Lines**). However, if you want to make playback of your score sustain without using these lines, use **~C64,127**. Switch the pedal off again with **~C64,0**. (Note that the sustain pedal control change is a switched function, i.e. it is either on or off.)

Soft pedal

Controller 67 can be used to achieve the effect of using the *una corda* pedal on a piano: to switch on the soft pedal, use **~C67,127**; to switch it off, use **~C67,0**. Note that this control change does not work on all MIDI devices.

Channel mode messages

You'll probably never need to use control changes 121-127 in Sibelius, but just in case:

- **~C121**: reset all controllers
- **~C122**: Local Control on/off
- **~C124-127**: Omni mode on/off, Mono/Poly mode

System messages

These don't need a channel, so the staff they are attached to only determines which MIDI device they are sent to.

System messages are split into three types: *system common messages*, *system real-time messages,* and *system exclusive messages*. Typically, only the latter are useful in Sibelius (the former two are largely connected with synchronizing MIDI with clock-based MIDI components, which is unsupported by Sibelius).

System exclusive messages are used to send data that is specific to the particular MIDI device you are using, and they may vary from device to device.

To enter system messages in your score:

- System exclusive: ~**X** *bytes* e.g. ~**Xh40, h00, hf7**. Note that normally you should put **hf7** at the end to terminate the system exclusive, unless you're going to follow it with a **D** command containing more data.
- System common/system real time: use raw data (below)
- Raw data (without any 'command' byte): ~**D** *bytes* e.g. ~**Dh40, h00, h7f**

Hiding MIDI messages

You probably don't want MIDI messages littered all over your score when you print it, so to hide them, simply switch off **Show text after tilde (~)** on the **Text** page of the **House Style** dialog box (**File** menu).

Only the ~ and the messages after it are hidden, so you can still read preceding instructions to the players such as 'mute' which are meant to be visible. MIDI messages still play when hidden. When you edit the text the ~ message reappears.

Dynamics

It's worth dwelling for a moment on the specific case of dynamics in the playback dictionary. When you enter a dynamic using Expression text (📖 **Text**), it derives its playback characteristics from the appropriate entry in the dictionary – e.g. ***fff*** equates to a MIDI velocity of 127. However, the actual velocity of the note will be determined by the **Balance** parameter in the **Sounds** dialog box (📖 **Sounds**), which is a scale factor applied to the note's velocity – e.g. if the **Balance** of a staff is set to **50%**, a ***fff*** dynamic in that staff will produce a MIDI velocity of 64.

But this isn't quite the end of the story – the actual playback velocity of a note also depends on the level of Espressivo and any articulations (such as accents) present.

Further information

If this topic hasn't exhausted your appetite for strings of letters and numbers, you can find more information on MIDI messages at the web site of the MIDI Manufacturers' Association (MMA), **www.midi.org**.