



Welcome to the Cantabile guides! Here you'll find everything you need to understand Cantabile.

Prefer a downloadable [PDF file](#)? Also, don't forget the [training videos](#) and if you've got questions the [community forum](#) is a great place to ask.

## Basics

- [Installation](#)
- [Getting Started](#)
- [Working with Plugins](#)
- [Ports and Routes](#)
- [Routing Diagrams](#)
- [Main Window Controls](#)
- [Tool Tips and Help Panels](#)
- [Upgrading from Version 2](#)

## Derek Cook's Guides

- [Cantabile Guide](#)
- [DMXIS Guide](#)
- [Using the Line 6 Helix with Cantabile](#)
- [Using Cantabile and the MuLab Plugin Together](#)

## Working with Audio

- [Configuring Audio Ports](#)
- [Mapping Channels between Audio Ports](#)

## Working with MIDI

- [MIDI Route Settings](#)
- [MIDI Filters](#)
- [MIDI Monitoring](#)
- [Global Transpose](#)

## Features

- [Audio Engine Options](#)
- [Bindings](#)
- [Controller Bar](#)
- [Countdown Timer](#)
- [Loopback Ports](#)
- [Media Players](#)
- [Metronome](#)
- [Managing Plugins](#)
- [MIDI Clock Synchronization](#)
- [Monitor Panel](#)
- [Morph and Randomize Tools](#)
- [Network Server](#)
- [Offline Render](#)
- [Onscreen Keyboard](#)
- [Pointer Lock](#)
- [Plugin and Parameter Editor](#)
- [Preset Models](#)
- [Quick Controller](#)
- [Racks](#)
- [Recording](#)
- [Song and Rack Properties](#)
- [Stream Deck Plugin](#)

## Performance Management

- [Song and Rack States](#)
- [Set Lists](#)
- [Ticker Bar](#)
- [Show Notes](#)
- [Live Mode](#)
- [Set List Verification and Printing](#)
- [Preventing Prompts to Save](#)

## Customizing Cantabile

- [Command Line Options](#)
- [Controller Defaults](#)
- [Custom Themes](#)
- [Customizing Shortcut Keys](#)
- [Expression Reference](#)
- [Functions](#)
- [Gain Control Curves](#)
- [Language Translations](#)
- [Multiple Configurations](#)
- [Rebranding](#)
- [Resource Folder](#)
- [SysEx Macros](#)
- [Variables](#)

## Diagnostics

- [Tuning for Reliable Glitch Free Performance](#)
- [Performance Profiler](#)
- [Diagnostic Options](#)
- [Settings Folder](#)
- [Prevent Memory Paging Options](#)
- [No Sound](#)

## Technology

- [Introduction to Computer Based Music](#)
- [Understanding 64-bit](#)
- [Understanding Multi-Core](#)
- [Understanding Cantabile's Performance Metrics](#)
- [jBridge Support](#)
- [Support for High-Resolution Displays](#)

## Licensing

- [Understanding Cantabile's Licensing System](#)
- [Understanding Cantabile's Subscription Model](#)
- [Installation on Multiple Machines](#)
- [My Serial Number Doesn't Work](#)

# Installing Cantabile

To install Cantabile just run the installer - there are no options.

The only pre-requisites are a compatible operating system.

Cantabile 4 (4300 and later):

- Windows 10 (Anniversary Update) or Windows 11 x64 (32-bit Windows no longer supported)
- Microsoft .NET 8.0.14
- Sound card with compatible ASIO driver or WASAPI drivers

Cantabile 4 (4150 and later):

- Windows 7 (SP1), Windows 8.1, Windows 10 (Anniversary Update) or Windows 11
- Microsoft .NET 6.0.24
- Sound card with compatible ASIO driver or WASAPI drivers

Cantabile 4 (before build 4150):

- Windows 7 (SP1), Windows 8.1, Windows 10 (Anniversary Update) or Windows 11
- Microsoft .NET 5.0.8
- Sound card with compatible ASIO driver or WASAPI drivers

Cantabile 3 (3500 Series):

- Windows 7 (SP1), Windows 8.1, Windows 10 (Anniversary Update) or Windows 11
- Microsoft .NET Framework 4.7.1
- Sound card with compatible ASIO driver or WASAPI drivers

Cantabile 3 (3200 Series):

- Windows XP SP3 or later
- Microsoft .NET Framework 4.0
- Recommended: Windows 7 or later
- Sound card with compatible ASIO driver or WASAPI drivers

For a complete walkthrough of installing and configuration Cantabile, see [this video](#).

## Updating to New Builds

To update to a newer build of Cantabile you don't need to uninstall the previously installed version - just run the new build's installer and everything will be taken care of automatically.

In general older builds of Cantabile *will* load files saved by newer builds however settings related to new features obviously won't be used by the older build.

There are a few exceptions to this and to be absolutely sure your current files will load in older builds it's a good idea to make backup copies of those files before installing the new build. If this is really important Cantabile will display a warning the first time the new build is run.

## Reverting to Older Builds

In most instances you can revert to an earlier build of Cantabile by running the installation program for the earlier build.

The one exception to this is when reverting from build 3694 (and later) to build 3693 (and earlier) - in this case you should uninstall Cantabile using Windows Add/Remove Programs before installing the older build.

The reason for this is build 3694 and later include additional configuration files that prevent the earlier builds from running. These files need to be removed for the earlier build to work correctly. Uninstalling Cantabile removes these files to create a clean directory before installing the older build.

Alternatively, if you're technically minded you can go to the Cantabile installation folder and manually remove any files matching \*.exe.config.

## Manually Installing Cantabile

If you want more control over the installation, you can use the portable .zip packages to manually install Cantabile. These files are available on the Release Notes ([v3](#) [v4](#)) page and contain a readme file with additional instructions and notes on their usage.

### Manually Installing .NET (Cantabile v4 since 4300)

From build 4300 onwards, Cantabile 4 requires .NET 8.0. Normally the installer will detect if the required version is already installed and if not will automatically download and install it for you. If the machine you're installing on doesn't have an internet connection this will fail and you'll need to install it manually yourself.

If you let the Cantabile installer download and install .NET, please be patient - it's takes a few minutes to install.

To manually install .NET 8 you need to install both the x64 versions found under the ".NET Desktop Runtime 8.0.x" heading on this page: [Microsoft .NET 6.0](#).

Alternatively, here are direct links to a recent version of the installers:

- [.NET 8 Desktop Runtime x64 Installer](#)

### Manually Installing .NET (Cantabile v4 since 4150)

From build 4150 onwards, Cantabile 4 requires .NET 6.0. Normally the installer will detect if the required version is already installed and if not will automatically download and install it for you. If the machine you're installing on doesn't have an internet connection this will fail and you'll need to install it manually yourself.

If you let the Cantabile installer download and install .NET, please be patient - it's takes a few minutes to install.

To manually install .NET 6 you need to install both the x86 and x64 versions found under the ".NET Desktop Runtime 6.0.x" heading on this page: [Microsoft .NET 6.0](#).

Alternatively, here are direct links to a recent version of the installers:

- [.NET 6 Desktop Runtime x64 Installer](#)
- [.NET 6 Desktop Runtime x86 Installer](#)

**IMPORTANT:** the installer checks that both the x86 and x64 editions are installed. Unless you manually install both, the installer will always attempt to install any missing editions.

### Manually Installing .NET (Cantabile v4 before 4150)

Cantabile 4 up to build 4064 requires .NET 5.0. Normally the installer will detect if the required version is already installed and if not will automatically download and install it for you. If the machine you're installing on doesn't have an internet connection this will fail and you'll need to install it manually yourself.

If you let the Cantabile installer download and install .NET, please be patient - it's takes a few minutes to install.

To manually install .NET 5 you need to install both the x86 and x64 versions found under the ".NET Desktop Runtime 5.0.x" heading on this page: [Microsoft .NET 5.0](#).

Alternatively, here are direct links to a recent version of the installers:

- [.NET 5 Desktop Runtime x64 Installer](#)
- [.NET 5 Desktop Runtime x86 Installer](#)

**IMPORTANT:** the installer checks that both the x86 and x64 editions are installed. Unless you manually install both, the installer will always attempt to install any missing editions.

## Manually Installing .NET (Cantabile v3)

Cantabile requires Microsoft .NET in order to run. Normally the installer will detect if the required version is already installed and if not will automatically download and install it for you. If the machine you're installing on doesn't have an internet connection this will fail and you'll need to install it manually yourself.

If you're not sure if you need this, run the Cantabile installer and it will notify you if the download is required before the setup starts. If it's required and you don't have an internet connection, cancel the Cantabile installer, install .NET using the link below and then re-run the Cantabile installer.

- Before Cantabile build 3500 - [Microsoft .NET 4.0](#)
- Cantabile build 3500 and later - [Microsoft .NET 4.7.1](#)

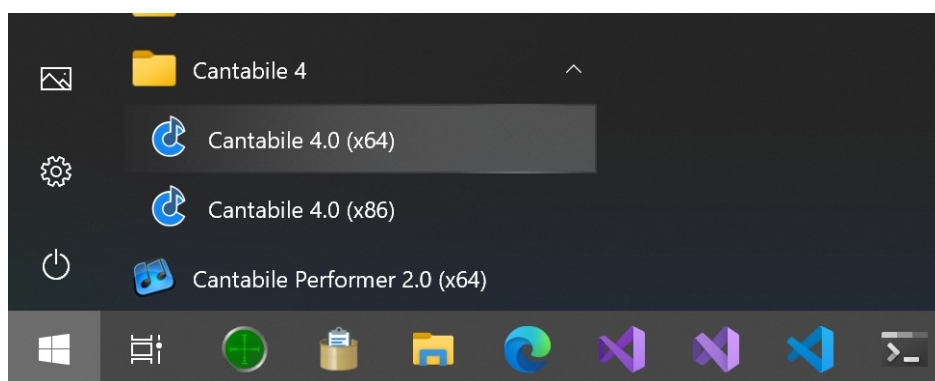
If you let the Cantabile installer download and install .NET, please be patient - it's takes a few minutes to install.

## Drivers and Plugins

Aside from Cantabile itself you'll also need a suitable audio driver, MIDI drivers and plugins installed. Configuring these is beyond the scope of this document however if you're new to computer based music, [this guide](#) might help you get started.

## X86 vs X64

Cantabile is available for two platforms - x86 and x64. The installer installs both editions and creates two shortcuts in your Windows start menu - one for each platform.



For a detailed explanation of x86 vs x64, [see here](#).

## Configuring Cantabile

Once you've installed Cantabile, see [Getting Started](#) for how to configure it for the first time.

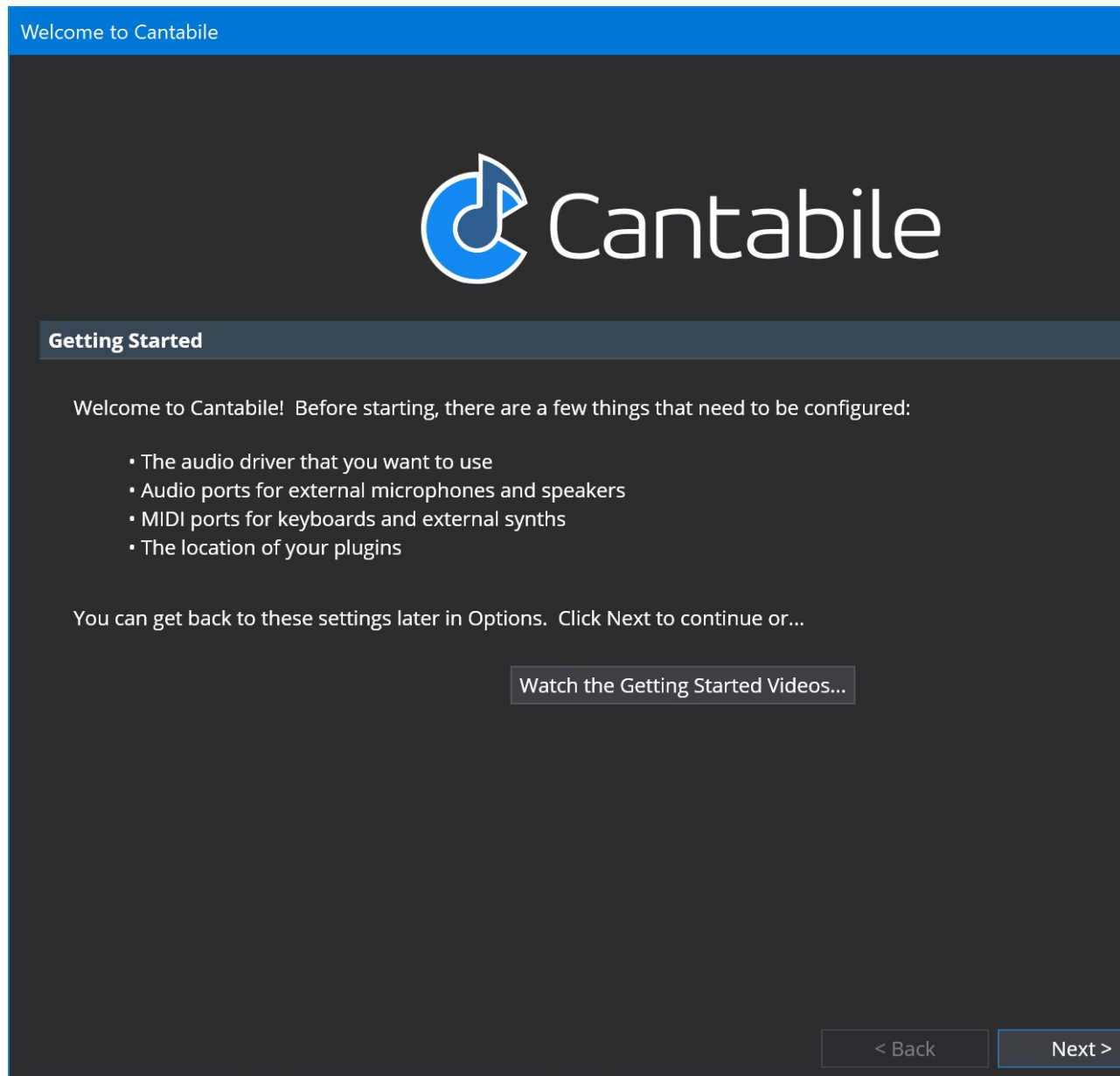
## Getting Started with Cantabile

This guide explains how to get started with Cantabile, including configuring an audio driver, MIDI devices, VST plugins. It assumes you're familiar with basic computer music terminology and concepts - if not please [read this first](#).

For a complete walkthrough of installing and configuration Cantabile, see [this video](#).



The first time you run Cantabile you'll be presented with a series of screens that will help you setup Cantabile for the first time.



## Audio Engine Options

On the second screen you'll be prompted for the audio driver to use as well as the buffer size and sample rate the audio engine should run at. Normally the default will suffice.

Note: if you select the Null audio driver it won't get any sound output - it's for MIDI only processing.

### Audio Driver

Audio Driver: ASIO - ASIO4ALL v2

Sample Rate: 44,100Hz

Buffer Size: 512

☐ Shared Mode (allows other programs to use same sound card)

☐ Double Buffered Audio

Buffer Duration: 11.61ms

Please choose the audio driver, sample rate and buffer size to use.

In most cases a sample rate of 44,100Hz and a buffer size of 256 provides a good balance between quality, performance and stability.

< Back

Next >

## Audio Ports

Next you'll need to configure some audio ports. Audio ports define the mapping between your song files and the physical hardware.

Cantabile will automatically create a default set of audio ports and take a guess at how you might like them mapped to the audio driver you selected in the previous step. You can accept these defaults, tweak the channel assignments or create a completely new set of ports.

At the very least you should have an output audio port with its channels assigned to output channels on the audio driver.

## Audio Ports

Name	Assignment	
<b>in: Main Microphone</b>		
Left	#1: HD Audio Mic input 1	
Right	#2: HD Audio Mic input 2	
<b>out: Main Speakers (Default)</b>		
Left	#1: HD Audio output 1	
Right	#2: HD Audio output 2	
<b>out: Metronome</b>		
Signal	#1: HD Audio output 1	

Showing assignments for ASIO - ASIO4ALL v2

Add



Edit

De

Here you can create audio ports that will be available to your songs  
and map them to the physical channels of your sound card.

(Tip: you can quickly create simple assignments by right clicking in the list above)

&lt; Back

Next &gt;

## MIDI Ports

Similarly to audio ports, MIDI ports provide a mapping between your song and physical MIDI devices.

Cantabile will create a couple of default MIDI ports for you but doesn't automatically map those ports to physical hardware. If you have external MIDI devices make sure you map them here by right clicking on a port and creating the assignment.

**MIDI Ports**

Name	Settings	
<input checked="" type="checkbox"/> in: Main Keyboard (Defa...	(Not assigned)	
<input checked="" type="checkbox"/> in: Onscreen Keyboard	Onscreen Keyboard	
<input checked="" type="checkbox"/> out: External Synth	(Not assigned)	
<input checked="" type="checkbox"/> out: Onscreen Keyboard	Onscreen Keyboard	

Add



Edit

Similarly to audio ports, here you can create MIDI ports and map them to your external MIDI devices. Right click to create simple assignments.

&lt; Back

Next &gt;

## Plugins Folder

Finally, Cantabile needs to know where your VST plugins are installed. You can also select a presets folder - any .fxb or .fxp files found in this folder will also be shown in the plugin selector.

Welcome to Cantabile

General

☐ Automatically open newly inserted plugins

VST Plugins

VST Plugin Folders:

☒ C:\Program Files\VstPlugins

☒ C:\Program Files\Steinberg\VstPlugins

☒ C:\Program Files\Common Files\Steinberg\VST2

☒ C:\Program Files\Common Files\VST3

Add

▼

Remove

Presets and Banks

Presets and banks folder:

Finally, please specify where your VST plugins are installed.

You can also optionally specify a folder that contains .fxb and .fxp preset files.  
These files will be made available for quick access when you insert a plugin.

< Back

Finish

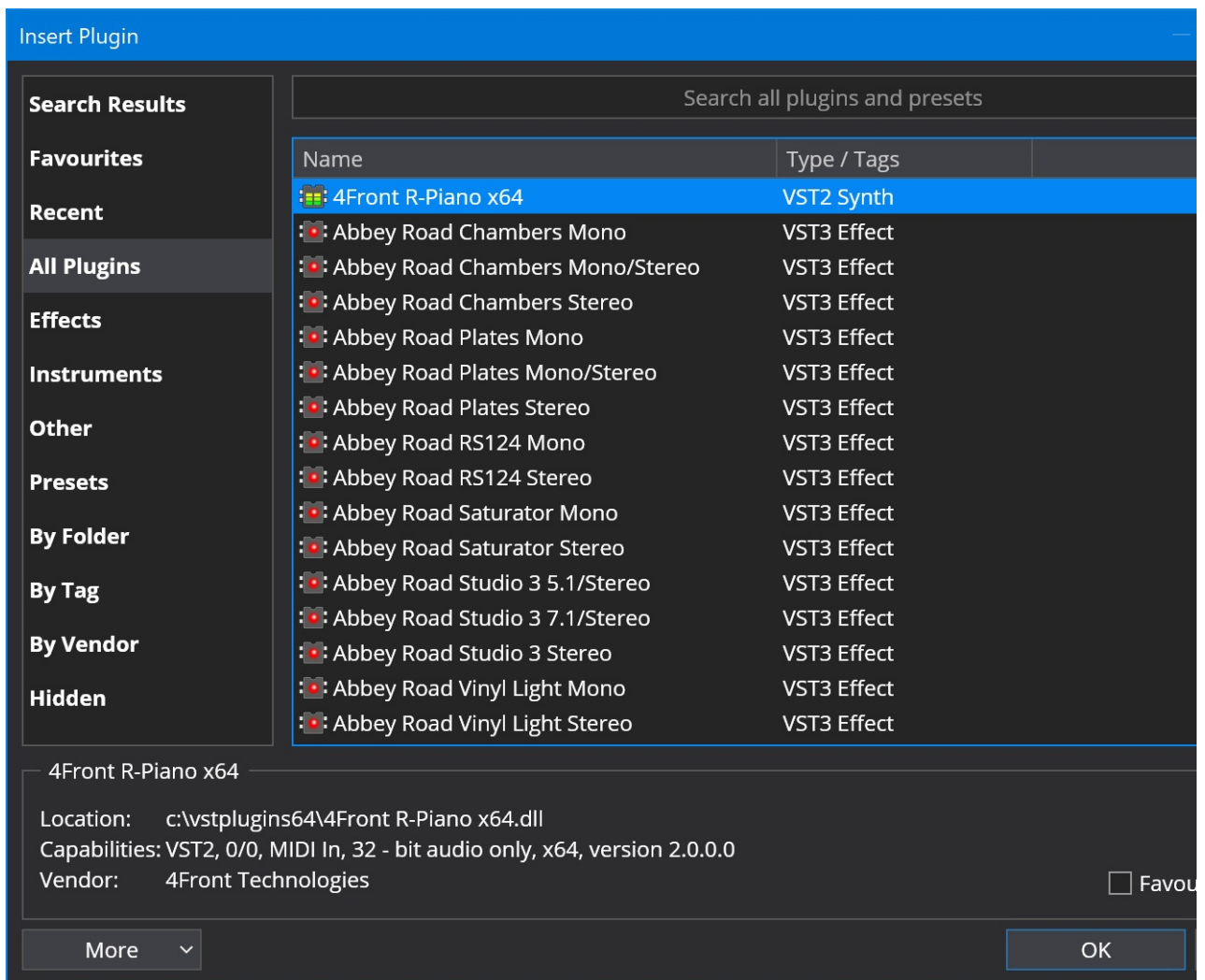
Click Finish and you're ready to starting using Cantabile.

## Working with Plugins

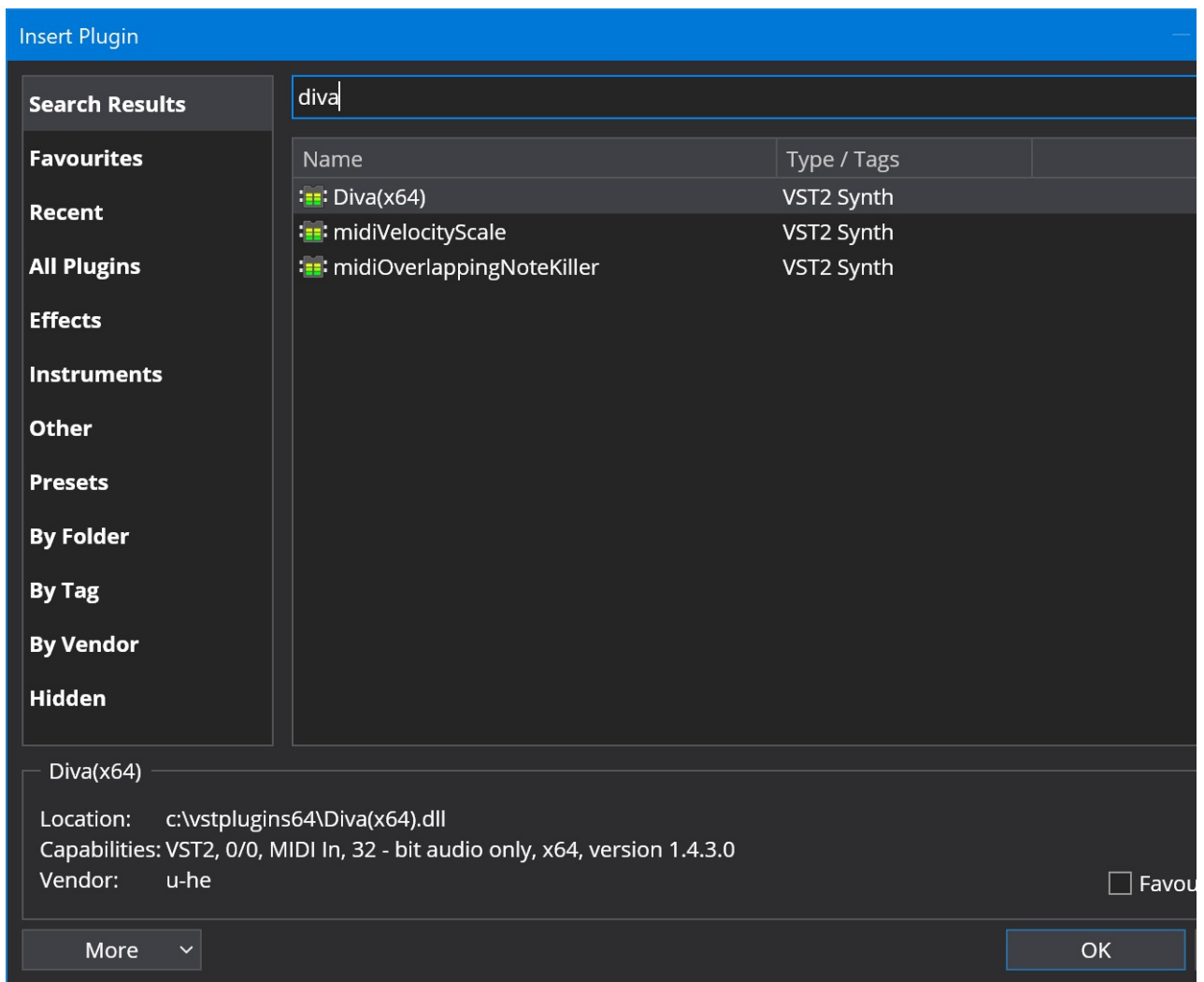
One of the most common tasks when using Cantabile is setting up plugins and the routes between them. This guide is a simple walk through for inserting a couple of plugins and connecting them together.

### Inserting a Plugin

To insert a plugin, either click the Add Plugin button in Cantabile's main window area, or choose Insert → Plugin from the main menu. You'll be presented with the plugin selector:



Locate the plugin you're after by browsing the categories on the left and the list on the right, or as a short cut just type the first few letters of the plugin's name and the list will be filtered down. eg:

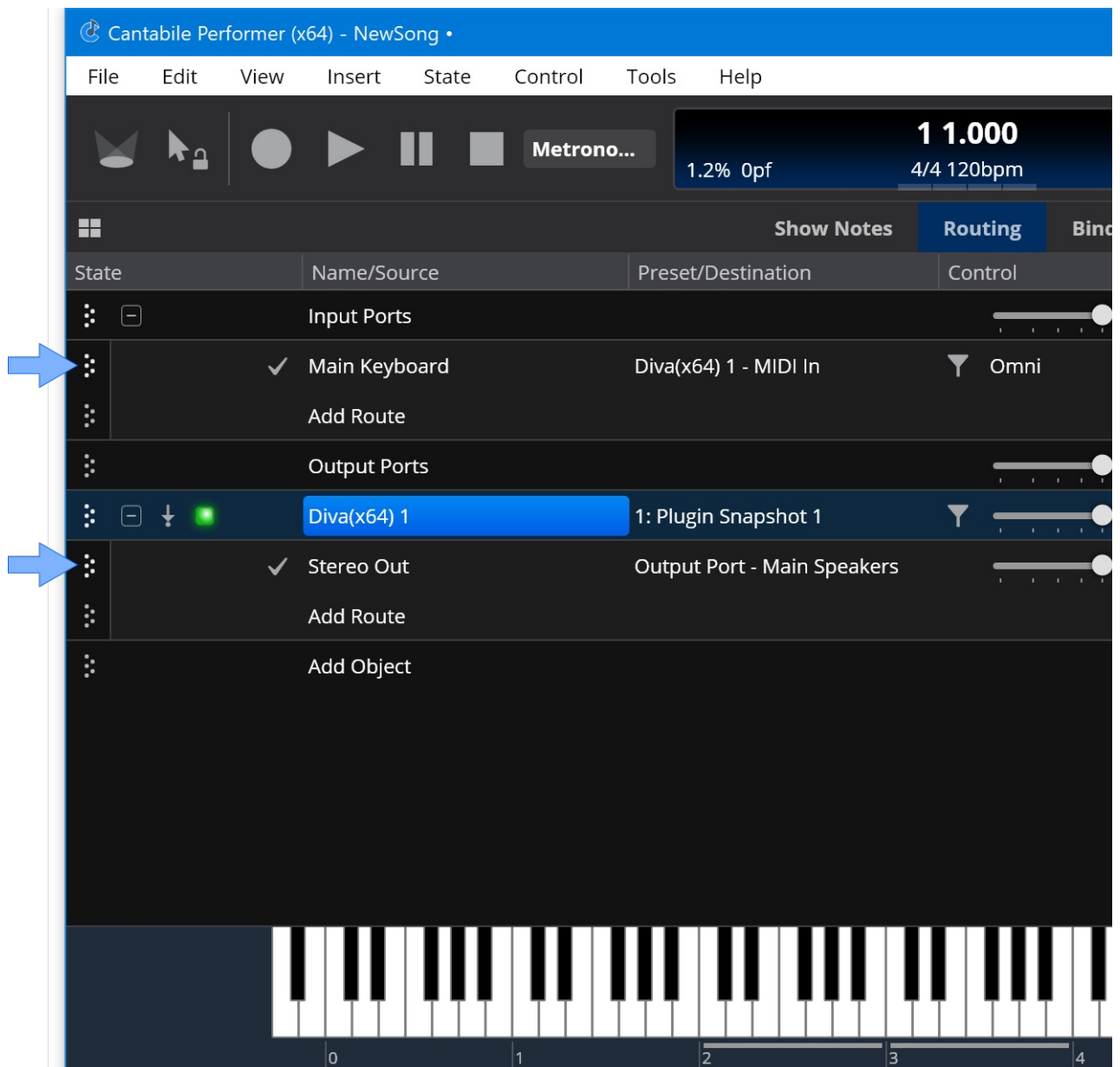


For more information about managing plugins including setting plugin tags, see [Managing Plugins](#).

## Automatic Routes

When a plugin is inserted, Cantabile automatically creates certain routes. In this case it created a MIDI route from the default MIDI input port (Main Keyboard) to the plugin and an audio route from the plugin to the default audio output port (Main Speakers).

The default ports can be configured in Options → MIDI Ports.S



## Editing Plugin Settings

To open a plugin's user interface either double click on it's name ("Ivory VST 1" in this case), or select it and choose Edit → Plugin Editor from the main menu. You can also edit the plugin's parameters using Cantabile's parameter editor by Alt+double clicking the plugin, or by choosing Edit → Plugin Parameters from the main menu.

In this example, once a keyset has been loaded into Ivory the plugin can be played from whatever MIDI device is mapped to the Main Keyboard port (in Options → MIDI Ports).

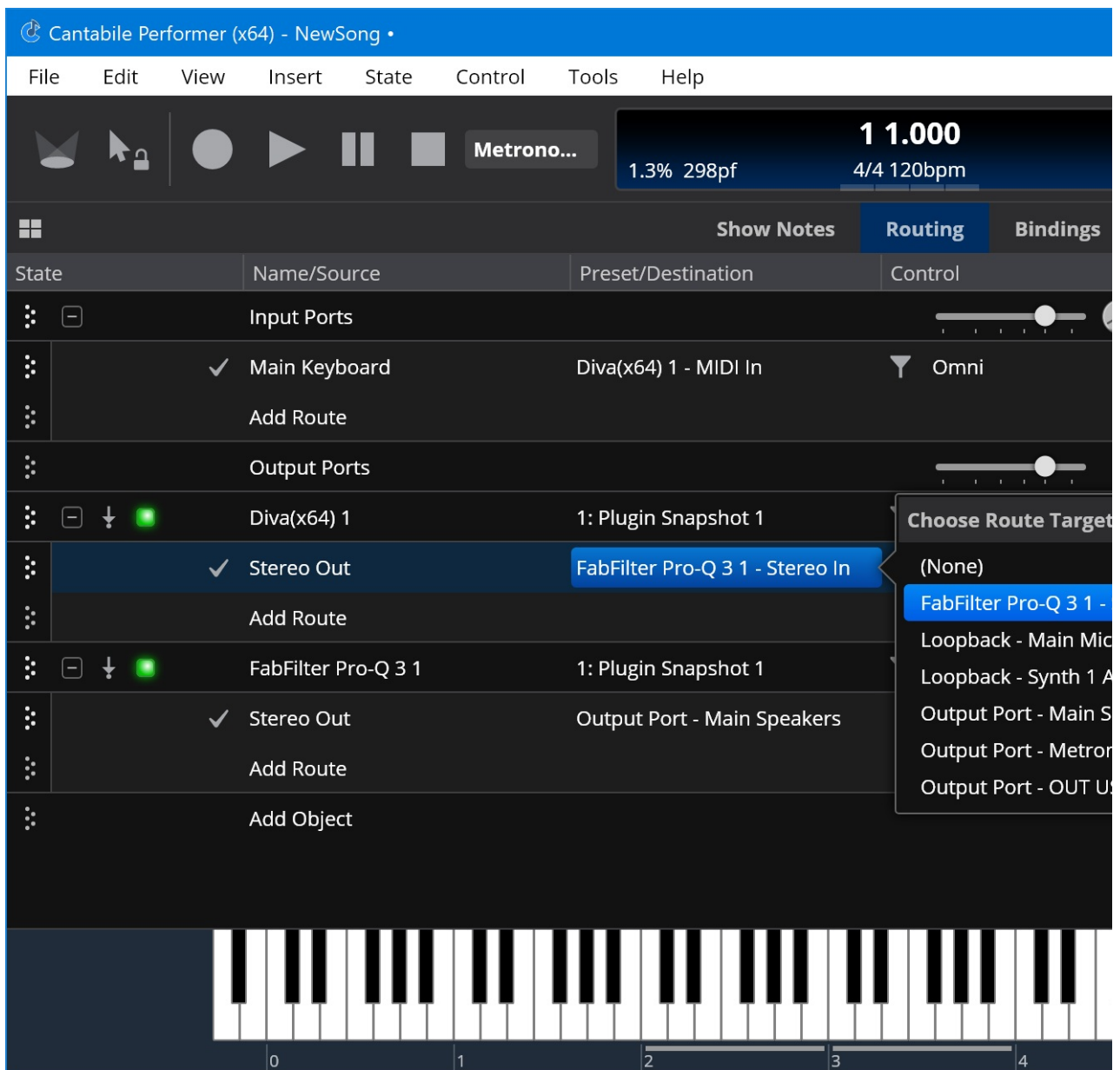
## Routing to a Second Plugin

Suppose now we want to send the output of Ivory through an effect. Using the same procedure above insert the effect plugin (in this case FabFilter). This time, Cantabile creates a route from the new plugin to the Main Speakers output port but doesn't connect any input routes to the plugin. This is deliberate to prevent accidentally creating feedback loops.



To route Diva through FabFilter, just change its audio output route to send to the plugin:

To route Diva through FabFilter, just change its audio output route to send to the plugin:

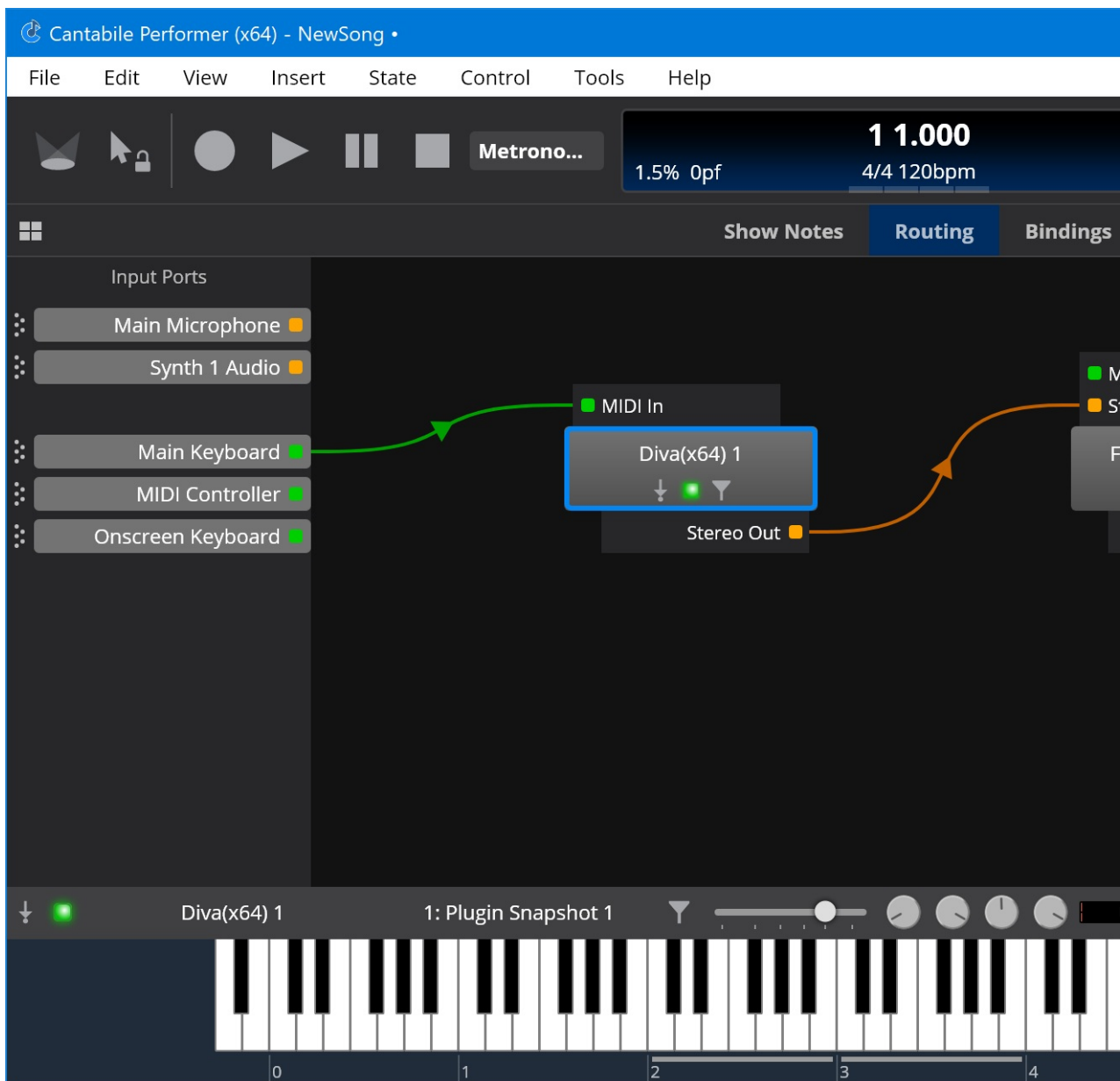


Now Diva's output will be processed by the effect plugin and then sent to the main speakers.

For more information see [Ports and Routes](#).

## Routing Diagrams

The above example, uses Cantabile Table Routing View. You can also use Routing Diagrams to more visually configure objects and routes. See [Routing Diagrams](#).



## Understanding Ports and Routes

Ports and routes are fundamental concepts in Cantabile that are used to connect objects together.

- Ports are connection points on an object. eg: an input port on a plugin, and output port to a sound card, an input port from a MIDI device etc...
- Routes represent the connections between ports. eg: a MIDI route from your keyboard input port to an input port on a plugin, or an audio route from an effect to your sound card.

A port is either an input port or an output port and a route always connect an output port to an input port. A port can accept multiple route connections eg: connecting two MIDI Keyboard ports to one plugin, or connecting the audio output of a plugin to two sets of output speakers.

It's convenient to think of a port as a socket and a route as a cable.

## Editing Routes

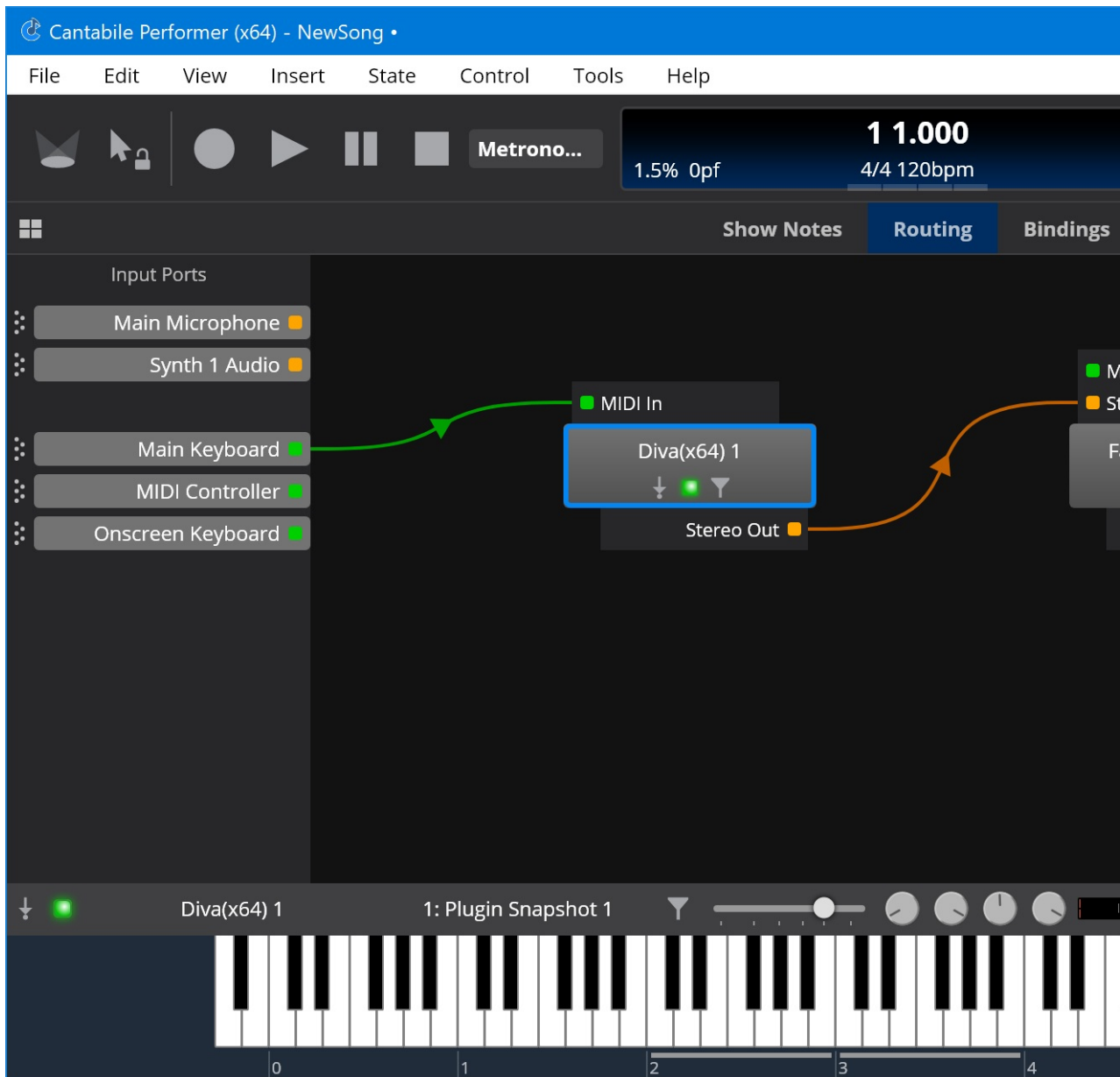
Cantabile's main Routing panel displays a list of objects (Input Ports, Plugins, Media Players, etc...) and each object is expandable to show the set of routes connecting the object to other objects.

The following example shows a Diva plugin with one route from its "Stereo Out" port connected to the "Main Speakers" output port.

⋮	[-] ↓ ●	Diva(x64) 1	1: Plugin Snapshot 1
⋮	✓	Stereo Out	Output Port - Main Speakers
⋮		Add Route	

- To create additional routes from an object, click the Add Route button.
- To edit an existing route click in the Source and Destination columns to change the connections
- To delete a route, select it and press the Delete key.
- To collapse an object to hide it's routes by clicking the downwards pointing triangle arrow at the far left of the plugin slot.

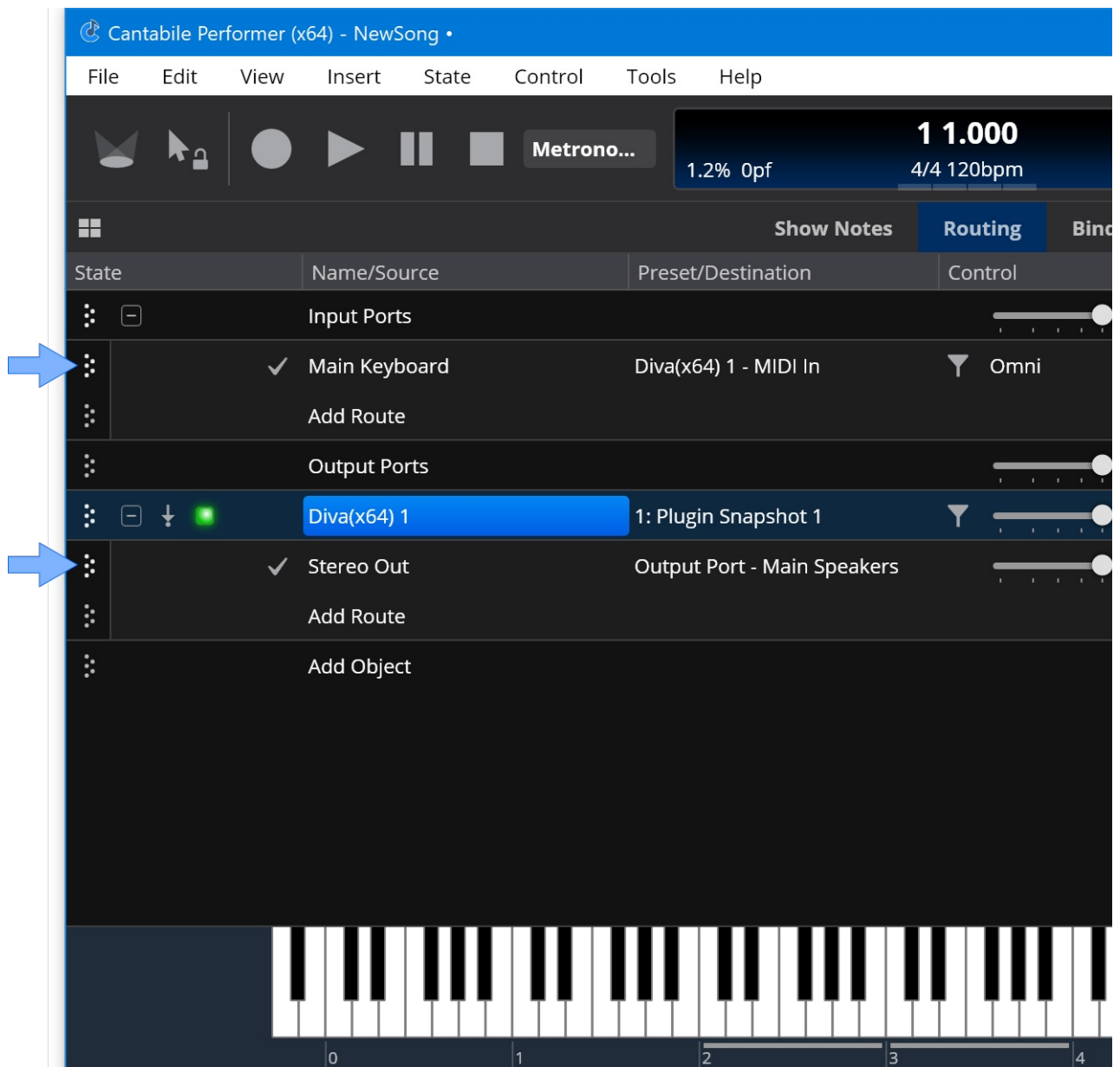
You can also use Routing Diagrams to more visually configure objects and routes. See [Routing Diagrams](#).



## Example

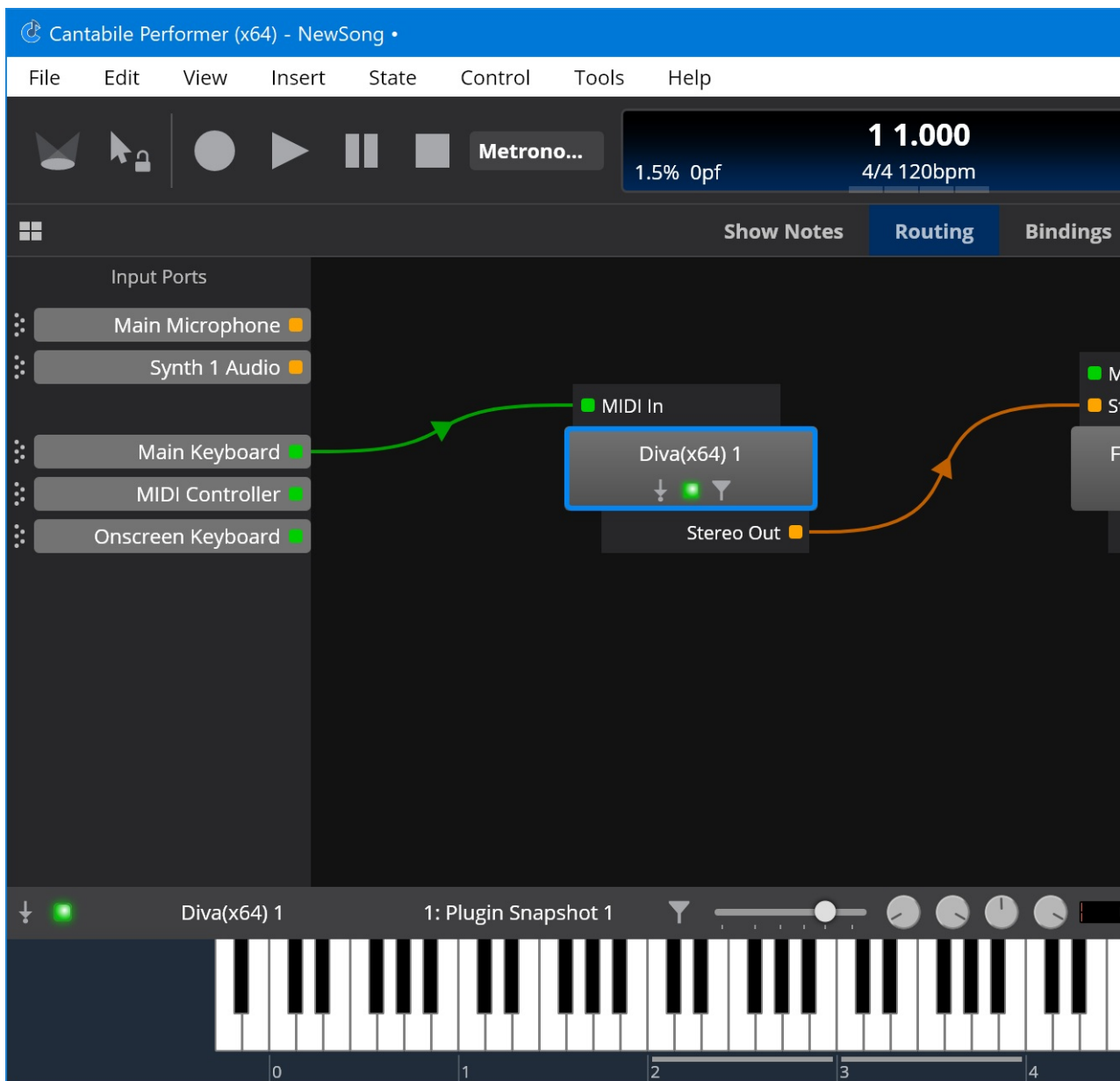
The following screen shot shows two routes:

- a MIDI route from the Main Keyboard to the Diva plugin and
- an audio route from Diva to the Main Speakers



## Routing Diagrams

Routing Diagrams provide a more visual way to view and edit the routes in a song or rack:

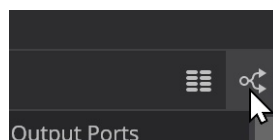


As can be seen in the above screen shot, the routing view is divided into three main areas:

- Input Ports on the Left
- Output Ports on the Right
- Objects and Routes in the Middle

## Switching Between Table to Routing Diagrams

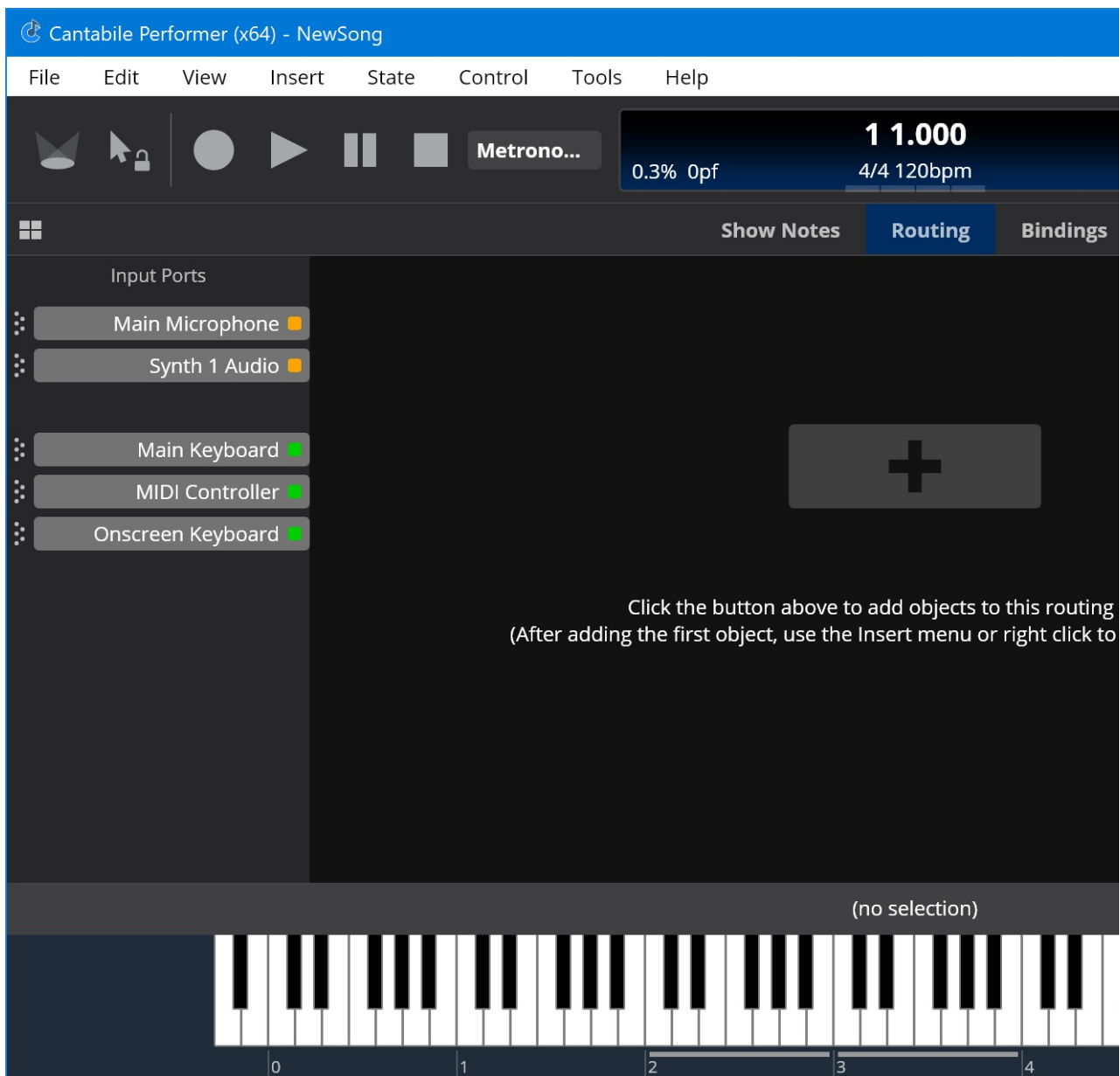
The table view and the routing diagram views both show the same set of objects and routes. You can switch between the two views using the view selector buttons at the top right hand corner of the screen:



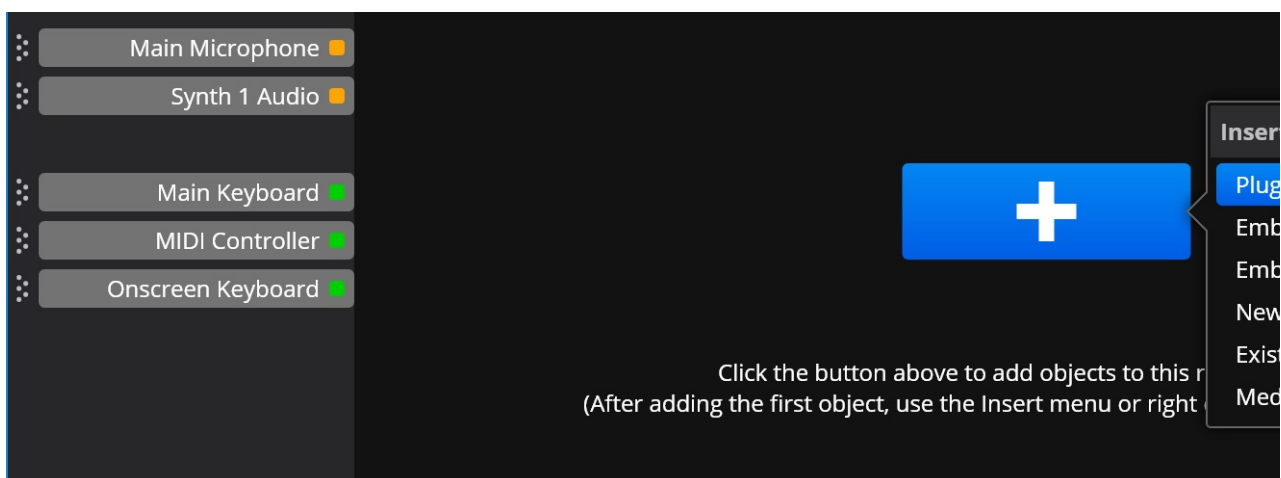
You can also switch views via the main **View** menu, or pressing **Ctrl+Shift+X** will toggle between the two views.

## Adding New Objects

The easiest way to add a new object to a routing diagrams is via the large "add" buttons. When the routing view is empty, a single centered add button always appears:



After you've added the first object, add buttons will appear in grid locations as you move the mouse around the screen:



You can also add objects by right clicking in a blank part of the diagram, or via the main **Insert** menu.

## Creating Routes

Routes are represented as wires that connect between ports.

To add a new route, click on any port and drag it to another port. Here a route is being added by dragging from the "Onscreen Keyboard" port and connecting it to the "MIDI In" port on a plugin:



Note that you can only connect MIDI ports to other MIDI ports and audio ports to audio ports.

## Editing Routes

To edit an existing route, select it by clicking on it and then use the round handles at either end to move the connection to a different port:



## Deleting Objects and Routes

Any object or route can be deleted by selecting it and pressing the **Delete** key.

## Moving Objects

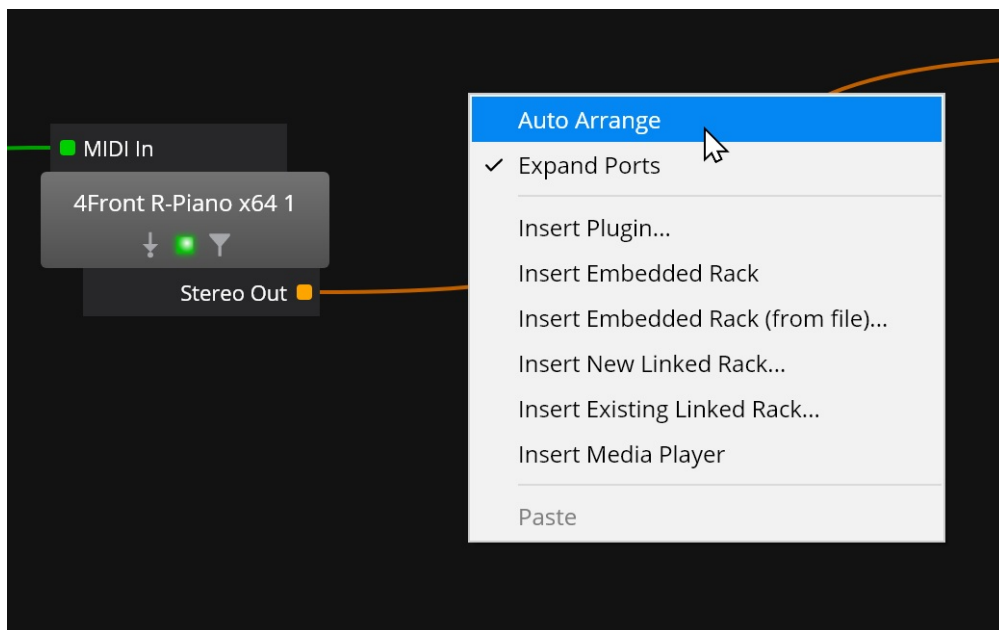
Often you'll want to reposition objects in the diagram to make it more visually pleasing. To do this, click on the object and drag it to the new location.

By default objects will "snap" to predefined grid locations in the diagram. You can suppress this snapping by holding the **Shift** key as you drag the object. (Press **Shift** after starting the drag otherwise you'll start a pan action - see below).

## Auto Arrange

Cantabile can automatically arrange the objects in a routing diagram to provide a pleasing layout. To auto arrange a routing diagram, choose **Auto Arrange** from the right click menu, or press **A**.





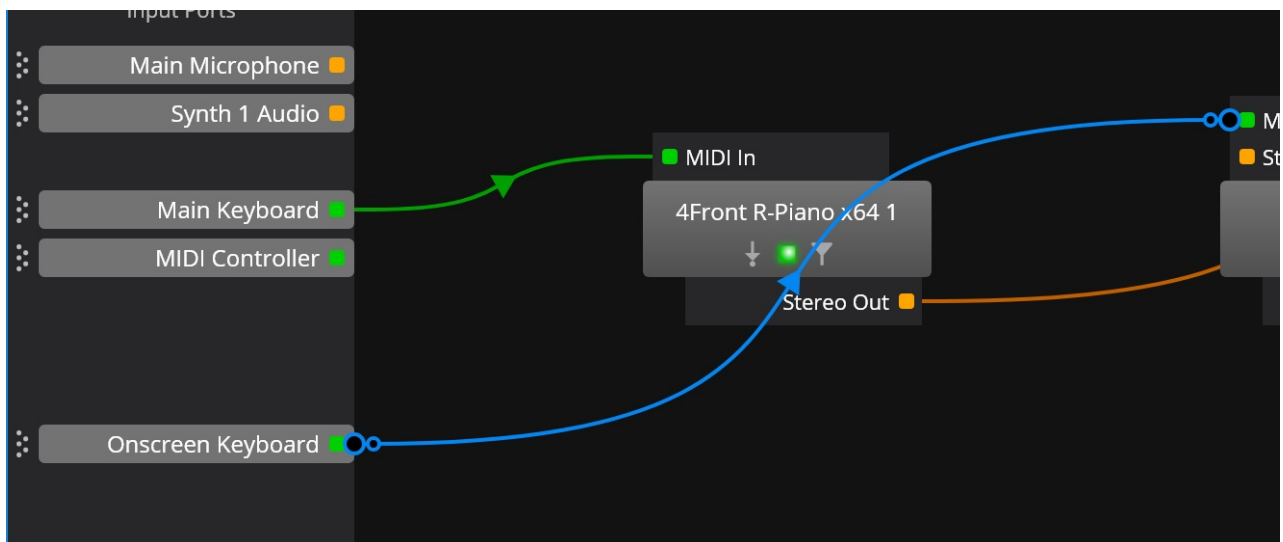
Auto arrange can be handy if you've added new objects using the Table View and Cantabile doesn't have a concept of where you'd like the object displayed in the diagram. In this case Cantabile will try to pick a reasonable position, but depending on how routes have been subsequently added the layout may not be ideal.

Note that Auto Arrange will reposition all objects. Don't use this command if you've already precisely laid out the diagram in a preferred manner.

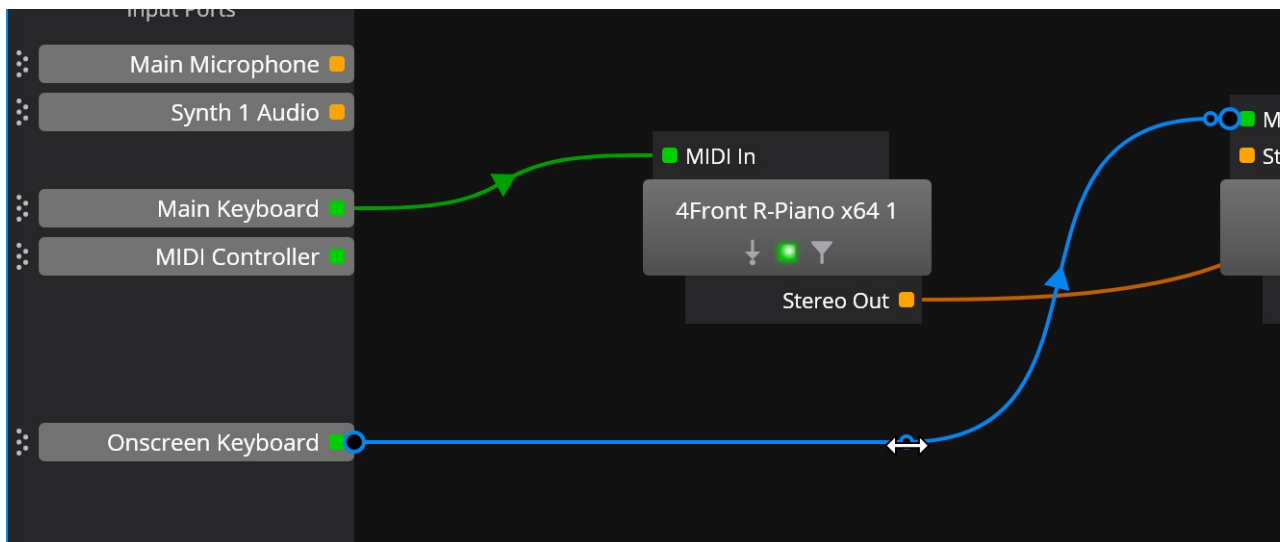
## Reshaping Routes

Sometimes you might like to reshape a route.

For example, this route overlaps with an unrelated plugin:

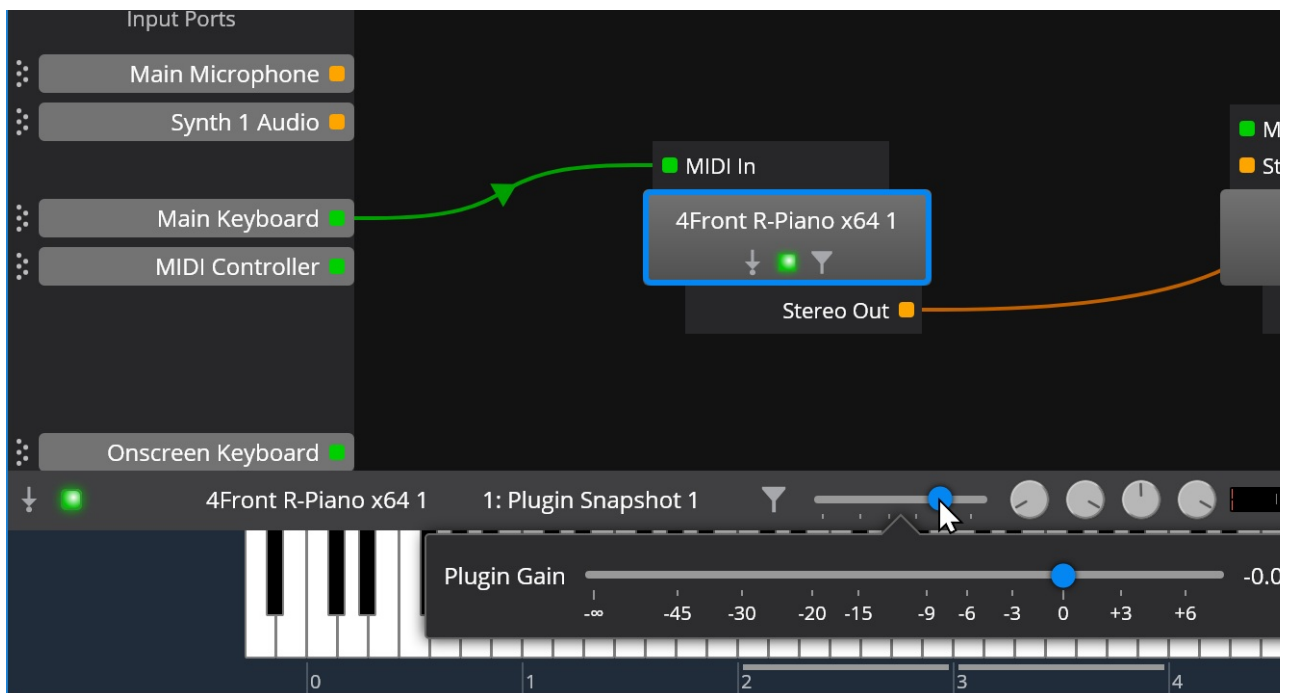


By using the smaller inset handle on the route, it can be reshaped to be more visually pleasing:



## Object and Route Settings

Along the bottom of the routing diagrams is a settings bar that shows settings for the currently selected object:



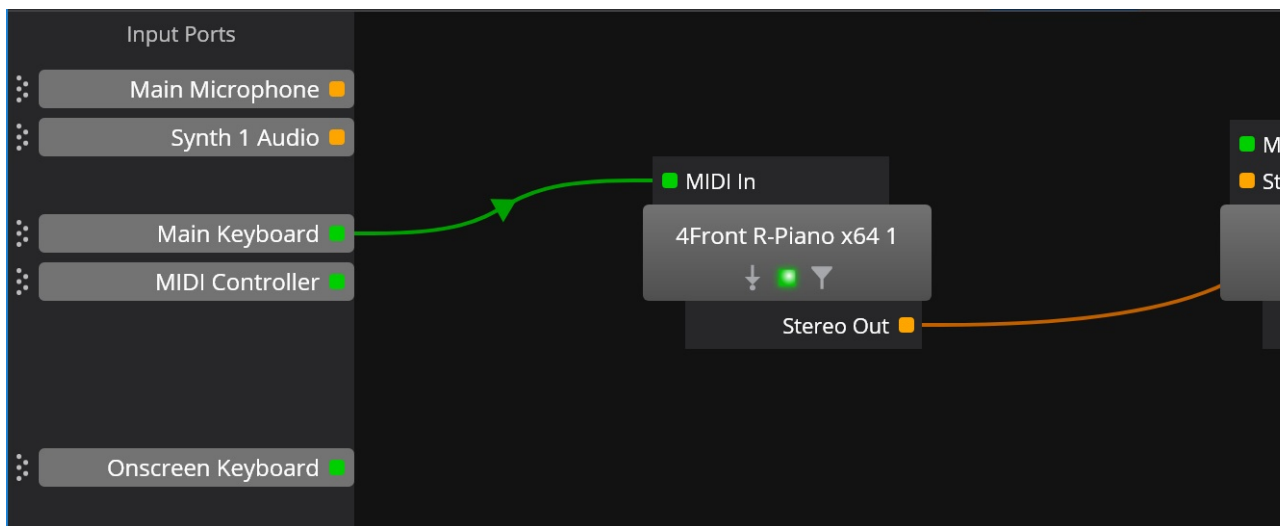
This bar very closely resembles the same settings in the table view.

## Hiding Object Ports

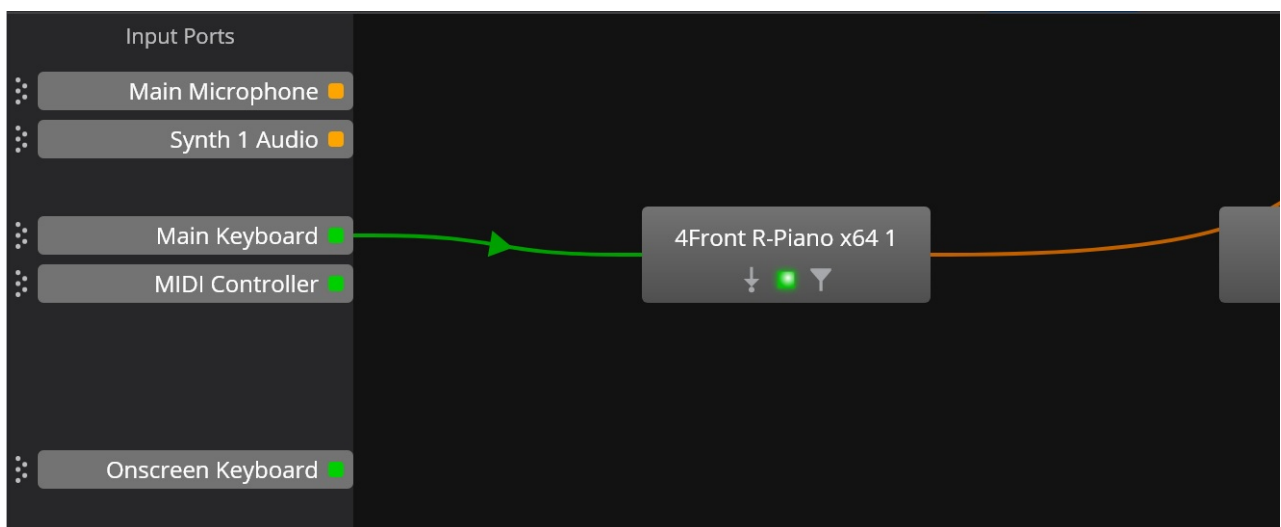
By default, the routing diagram shows the ports on each object. While this provides the most detail and is best when setting up a song, sometimes it can make the diagram appear more complex than it needs to be.

For a simpler view, press the **Tab** to toggle object ports on and off.

Here's a routing diagram with ports shown:



Here's the same diagram ports turned hidden:



While the ports are hidden, you can still interact with them as they temporarily appear when the mouse is hovered over an object.

## Selecting Multiple Objects

You can select multiple objects by:

- holding the **Ctrl** while clicking on each object or route
- clicking in a blank area of the diagram and dragging a rectangle around the desired objects (this will only select objects, not routes)

Multiple selection can be used to delete multiple objects at once, or to move multiple objects as a group.

## Zooming

Routing diagrams can be zoomed in/out as follows:

- Using the mouse wheel while pressing the **Ctrl** key
- Pressing **Ctrl+Up** and **Ctrl+Down**
- Using the **View** → **Zoom** menu
- On touch screens with pinch gestures

You can reset to the default zoom level by pressing **Ctrl+1**.

## Scrolling

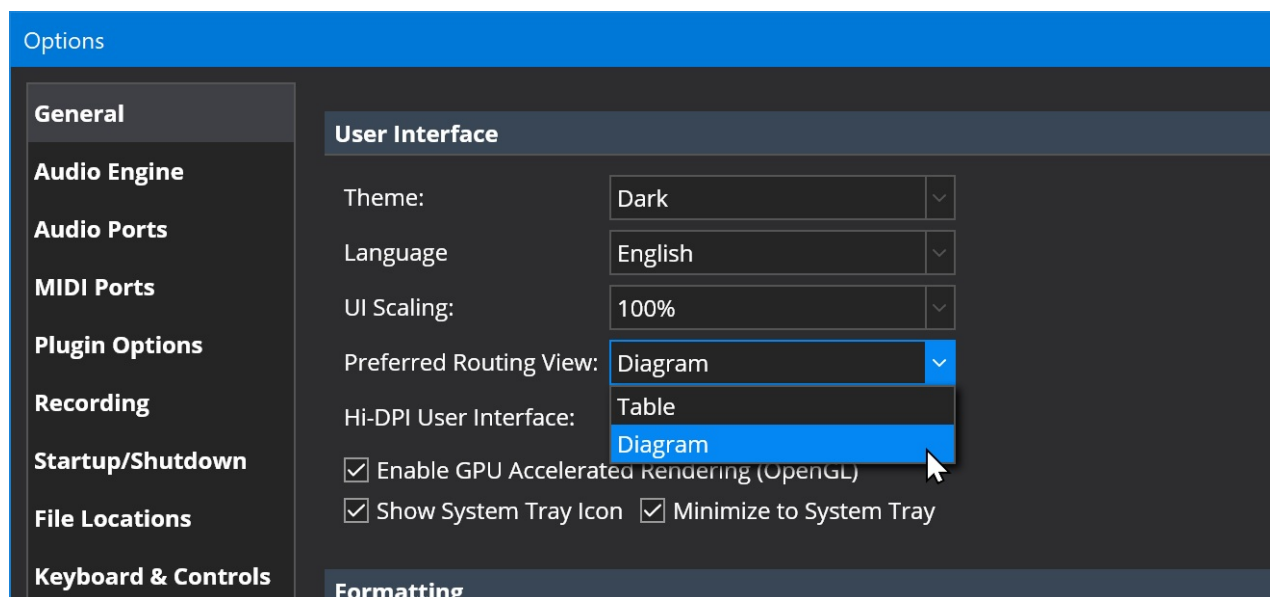
Scrolling (aka panning) the routing diagrams is supported by:

- Using the scroll bars

- Using the mouse wheel for vertical scrolling or hold **Shift** for horizontal scrolling
- Some track pads support both horizontal and vertical scrolling by two finger drag
- **Shift**+click and drag

## Preferred Routing View

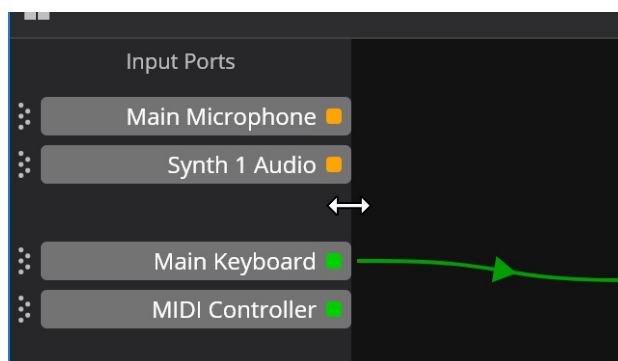
You can set your preferred routing view in Options → General → User Interface:



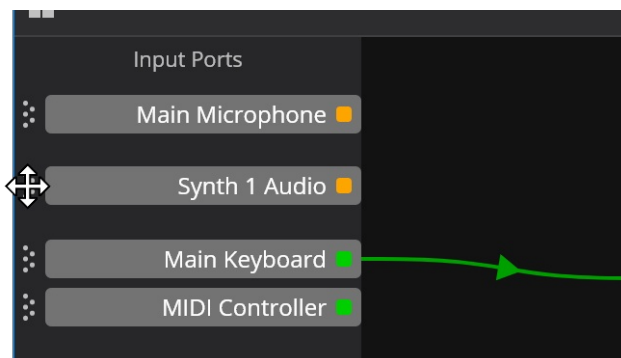
The preferred view is used to select which view (table or diagram) should be used for new songs and rack (and songs and racks created in versions of Cantabile saved before routing diagrams were introduced). Once a song or rack has been saved, it will store the current view and revert to that mode when the song is next open.

## Input and Output Port Area

The input and output port areas can be resized by clicking and dragging on their border:

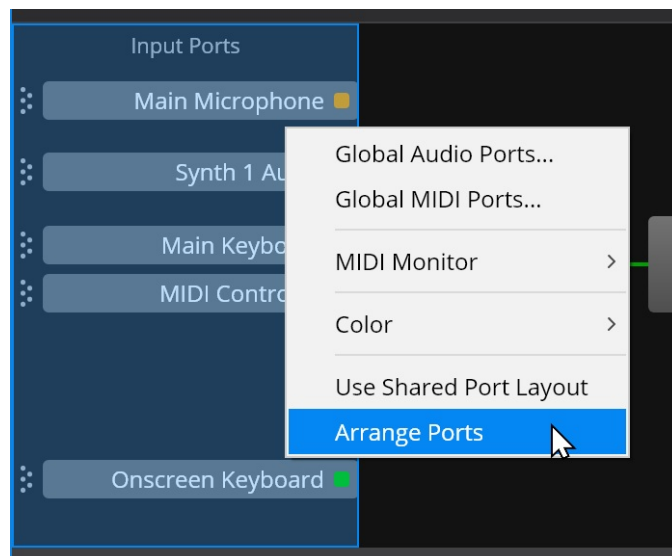


The input and output ports can be re-ordered and repositioned using the little gripper handles:



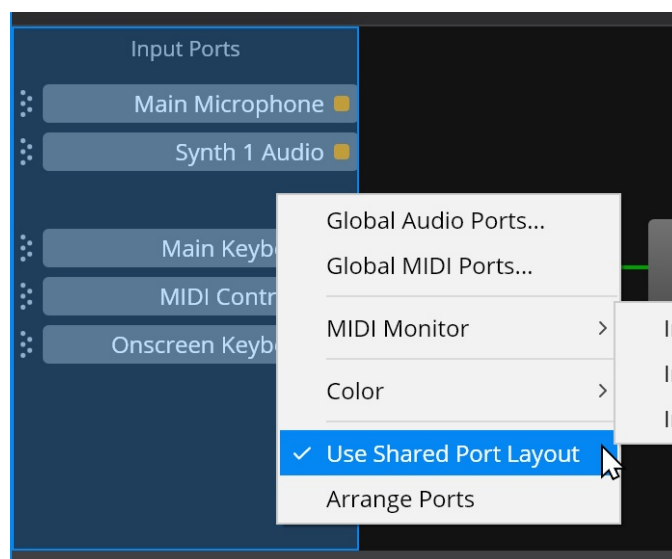
Tip: Holding the Ctrl key while dragging ports enters "push/pull" mode that makes it easier to insert and remove gaps between ports.

Right clicking on the input or output area provides commands for working with ports and the ability to auto-arrange the ports:



## Shared Port Layout

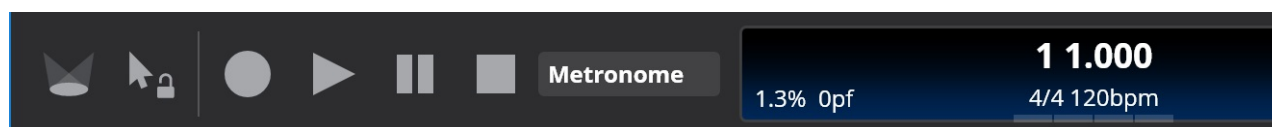
To use the same input/output port layout for all songs and racks, you can enable the option "Use Shared Port Layout". This option is available on a per-song/per-rack basis and all racks and songs with this option enabled will use the same port layout.



## Main Window Controls and Settings

The page gives an overview of all the controls and settings on Cantabile's main window.

### Main Toolbar

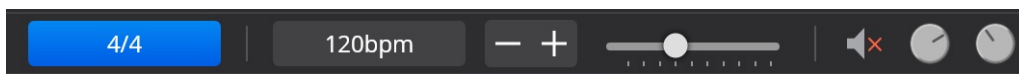


- Live Mode Button - switch to [Live Mode](#) (Cantabile Performer only)
- Pointer Lock - Disabled mouse and touch control for most settings to prevent accidental adjustment
- Record Button - starts and stops the recorder. Right click to configure Auto Record. See [Recording](#). (Cantabile Solo and Performer only)
- Master Transport Controls - Play, Pause, Stop
- Master Transport Selector - choose whether the metronome, MIDI clock, or a media player is the master transport for timing information.

- Status panel displaying the current transport position, time signature, tempo, beat indicators and a [load indicator](#).
- Tap Tempo Button - quickly set the tempo by tapping. Can also flash in time with tempo (right click to configure)
- Metronome Button - shows and hides the [metronome](#) toolbar
- Countdown Timer - a simple countdown timer for timing sets, practice sessions etc...
- Global Input and Output Gain - saved globally, not with the song
- Panic Button - sends note off and controller reset events to all plugins and MIDI output ports.
- Audio engine Power Button - glowing green indicates the engine is running.

## Metronome Toolbar

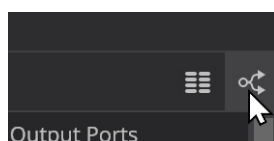
Clicking the Metronome button on the main toolbar shows and hides the metronome toolbar:



See [here](#) for more information on the metronome.

## Table/Diagram Routing View Switch

To the top right of the main window when on the Routing tab, there's a switch to select between the routing table view and routing diagram view:



See [Routing Diagrams](#) for more.

## Input Ports Slot



- Song level input gain control
- Song level transpose settings
- Audio input level meter
- MIDI input activity indicator

## Output Ports Slot



- Song level output gain control
- Audio output level indicator
- MIDI output activity indicator

## MIDI Route Slots



- The little note symbol indicates this is a MIDI route
- Enabled on/off selector
- A source port (ie: "Main Keyboard")
- A target object and port (ie: "Ivory VST 1 - MIDI In")
- MIDI Filter Settings (green indicator shows there are filters active on this route)
- Route settings display and control (ie: "Omni"), click to display MIDI route settings
- MIDI activity indicator that indicates when this route is receiving and/or sending MIDI

## Audio Route Slots



- The little wave form symbol indicates this is an audio route
- Enabled on/off selector
- A source port
- A target object and port

- A gain setting to control the signal level sent
- A level meter showing the audio level being sent (ie: after the gain setting is applied)

## Plugin Slots



- Expand/Collapse Arrow - click to expand or collapse the plugin to show routes from this plugin
- Bypass Mode - Bypasses all input audio directly to output (same as setting the wet/dry knob to completely dry)
- Run/Suspend Mode Indicator - lights up green when the plugin is running, orange when suspended and grey if unloaded.
- Plugin Name - Click to select the plugin, or right click for context menu.
- Preset Name - Name of the currently selected plugin preset, click to change.
- MIDI Filter Settings (green indicator shows there are filters active on this route)
- Gain Control - Controls the gain level of output signals from the plugin
- Wet/Dry Knob - Controls the mixing of input with output signals to control the amount of audio effect
- Balance Knob - Controls left/right balance or pan.
- Fade Knob - Controls front/rear balance or pan.
- Output Level Meter - Shows the signal level output from the plugin after gain and wet/dry controls have been applied
- MIDI Activity Indicators - Lights up when the plugin is receiving and/or sending MIDI events.

For more information see [Working with Plugins](#).

## Media Player Slots

(Cantabile Solo and Performer Only)



- Expand/Collapse Arrow - click to expand or collapse the media player to show routes.
- Transport Buttons - play and pause buttons allow control of this media player even when it's not the master transport
- Media Player Name - click to select the media player, or right click for context menu.
- File Selector - choose which file from the preconfigured playlist should be played.
- Gain Control - Controls the gain level of output signals from the plugin.
- Speed Knob - Adjusts the playback speed of this media player.
- Balance Knob - Controls left/right balance or pan.
- Fade Knob - Controls front/rear balance or pan.
- Output Level Meter - Shows the signal level output from the media player after gain and wet/dry controls have been applied.
- MIDI Input Indicator - Lights up when the media player is sending MIDI events.
- Play Position Percentage - displays the current play position as a percentage of the entire file length.

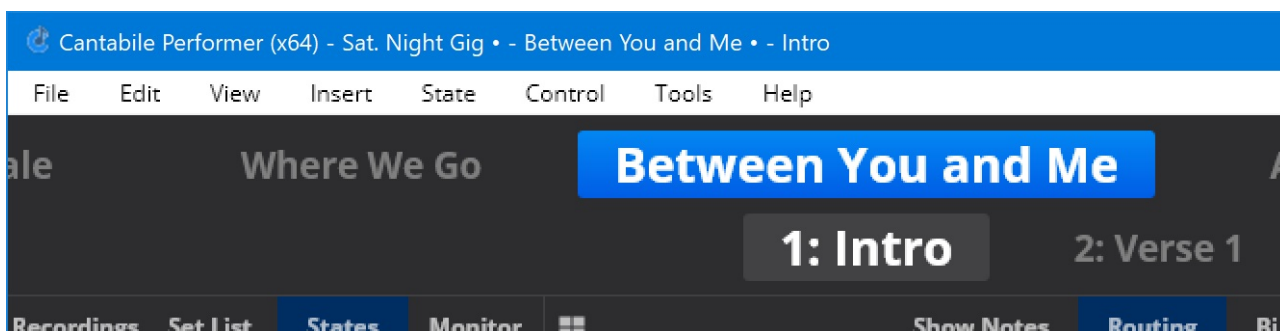
For more information see [Media Players](#).

## Rack Slots



- Expand/Collapse Arrow - click to expand or collapse the rack to show routes.
- Rack Icon
- Run/Suspend Mode Indicator - lights up green when the plugin is running, orange when suspended.
- Rack Name - Click to select the rack, or right click for context menu.
- Rack State Selector - Click to select a different rack state (Cantabile Performer Only)
- Rack Gain Setting - Controls the gain level of audio output signals from the rack.
- Output Level Meter - Shows the signal level output from the rack after gain and wet/dry controls have been applied
- MIDI Activity Indicators - Lights up when the plugin is receiving and/or sending MIDI events.

## Ticker Bar



See [Ticker Bar](#). (Cantabile Performer Only)

## Controller Bar



See [Controller Bar](#). (Cantabile Performer Only)

## Onscreen Keyboard



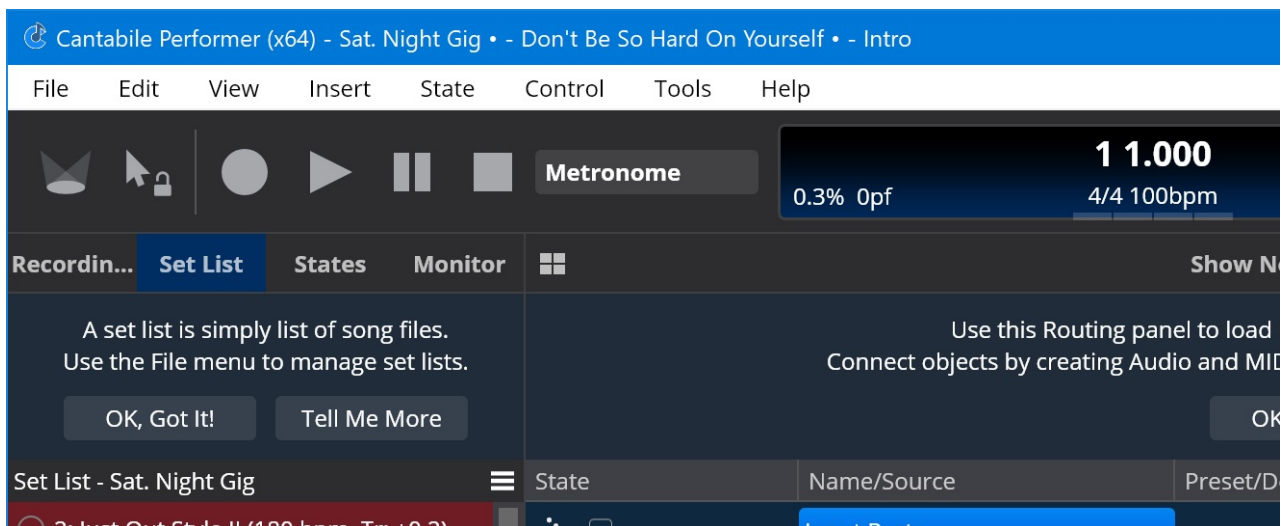
See [here](#) for how to use the on-screen keyboard.

## Tool Tips and Help Panels

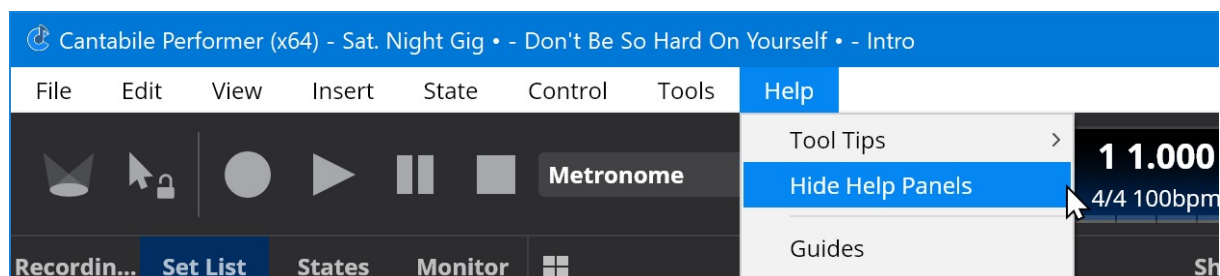
Cantabile has a couple of in-app features to help get you get up to speed.

### Help Panels

Help panels display a little panel at the top of the various work areas in Cantabile explaining what they do, a link to further information about that panel ("Tell Me More") and an button to dismiss the panel ("OK, Got it").



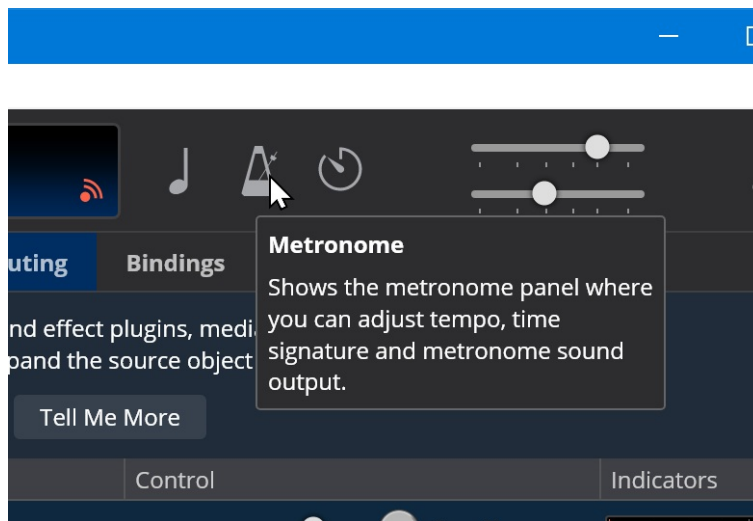
Once you're comfortable working with Cantabile you can dismiss all the panels (or re-show them all) using the Help → Hide Help Panels command:





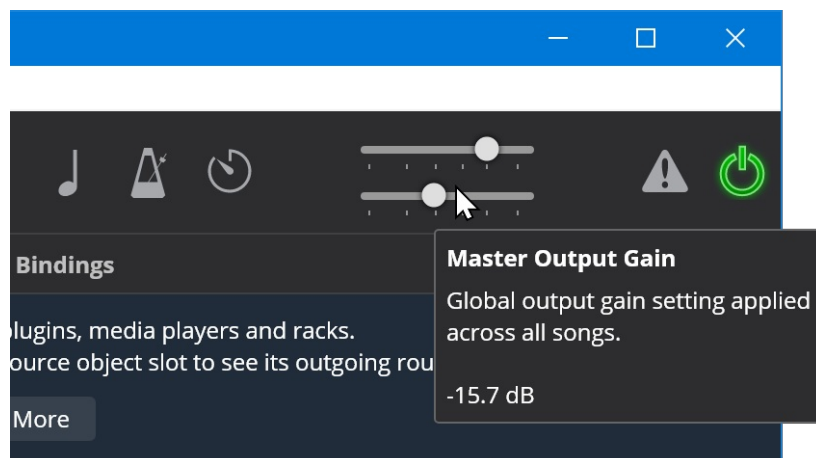
## Tooltips

Tooltips are another great way to learn what all of Cantabile's settings and buttons do:

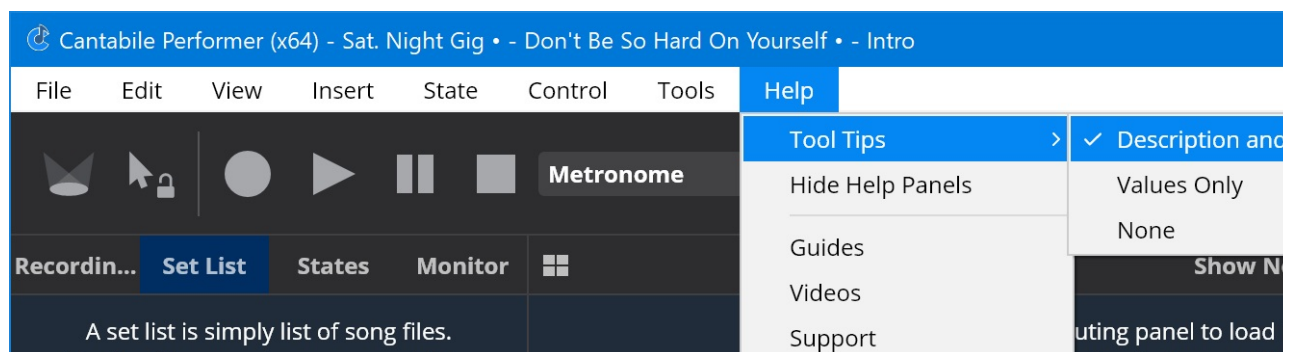


Some tooltips (mostly those on sliders and knobs) can also display the current value.

eg: note the 15.7 db shown in this tooltip:



The tooltip setting menu let you control whether you want to see regular tools with the value, or just the value:



## Upgrading from Cantabile 2

This guide explains how to upgrade Cantabile 2 session files and set lists.

**IMPORTANT:** Support for upgrading files from Cantabile 2 was removed in Cantabile 3 build 3677. If you need to upgrade files from Cantabile 2, you will need to:

1. Install build 3676 ([download](#))
2. Follow the steps below and convert all your sessions and set lists to the v3 format

3. Install the latest version Cantabile

## Background

Cantabile 3 (and later) introduces several fundamental conceptual changes over Cantabile 2. While in general these changes greatly improve Cantabile's usability and functionality they present a challenge when upgrading old session and set list files.

This guide gives an overview of these conceptual changes and provides tips for how to best approach upgrading.

## Before You Convert Your Version 2 Files

Before upgrading your Cantabile 2 session files you should make sure you have a back up of your old files. Although Cantabile 3 doesn't overwrite your old session files, backing up is always a good idea.

Also, it's highly recommended that you spend some time learning the new concepts in Cantabile 3. In particular you should be familiar with set lists, songs, racks and the new audio and MIDI routing capabilities of Cantabile.

Finally, you should make sure all the plugins used in your old files are known to Cantabile 3 - check your VST path and make sure you've recently run a plugin scan.

## Upgrading Session to Song Files

In Cantabile 3, session files have been renamed to song files.

To upgrade a Cantabile 2 session file, open it using the File Open command. You'll be prompted for where to save the converted file.

After conversion the file will be opened but will probably need adjustment and review before it will work correctly. (See sections below)

At the very least, the upgrade process will bring across:

- All plugins and plugin state

## Upgrading Set List Files

In Cantabile 2, each entry in a set list consists of a song name, a session file and a sub-session name. This has been simplified in Cantabile 3 and each entry is now just the name of a song file.

When upgrading set lists, Cantabile performs the following actions:

- The original session file is converted to a linked rack (unless a rack of the same name already exists)
- A new song file is created that references the linked rack
- The song file is saved using song's name from the original set list entry
- The song file sets the linked rack's state to the sub-session name from the set list
- The new set list is created referencing the new song files

You'll notice that conversion of song files is slightly different here compared to upgrading a single session file in that sessions are converted to racks, not songs.

The other side effect of this is that if you have a set list where many songs reference one session file then the resulting set list will consist of a list of song files all referencing the same linked rack.

If you're using set lists in Cantabile 2 you should in general upgrade the set lists rather than individual sessions.

## Handling MIDI Device Naming

In Cantabile 2, session files directly reference the names of MIDI devices (eg: Edirol PCR-M88). In Cantabile 3 devices are mapped through named ports which you configure in settings (eg: Main Keyboard)

In upgrading a version 2 file, Cantabile will try to map device names to ports by looking for a port with a direct one to one mapping to that device.

It will also however look at a port's alias names - so if you know you want to map a particular device name to a particular port specify the device name in the port's alias field.

Finally, if a rack in the original file was set to accept input from any input device, the upgraded session will create a routing from a device named "Any" - which probably won't exist but can easily be resolved - see below.

After upgrading a session file you should check all the referenced MIDI device names. Missing or unmatched devices will be shown as a broken routes and can be easily fixed by either:

- Changing the route to reference an available device,
- Creating a new port with the referenced name, or
- Setting an alias names on existing ports to automatically map old device names to the new ports. (See Options → MIDI Ports → Edit)

Aliases are a good option when you have many songs that reference a device that has a new name in Cantabile 3. After matching to an alias name, Cantabile will rename the route to use the actual port name. Once all your sessions have been upgraded you can remove the alias name.

## Audio Channel Mapping

In Cantabile, all audio channels were mapped through the master bus. Cantabile 3 no longer has a concept of master bus and uses named ports instead.

When upgrading a version 2 file, Cantabile will create ports on all plugins, media players and racks with the same number of channels as the master bus in the upgraded song file. These ports will be named "Audio In" and "Audio Out".

If the old session used a stereo master bus and no complicated audio channel mapping most sessions should just upgrade.

If your old session used a more complicated multi-channel master bus or audio routings you'll almost certainly need to revisit the audio channel mappings. You should check the each plugin, racks and media player by selecting them one by one and using the Audio Ports command in the Edit menu.

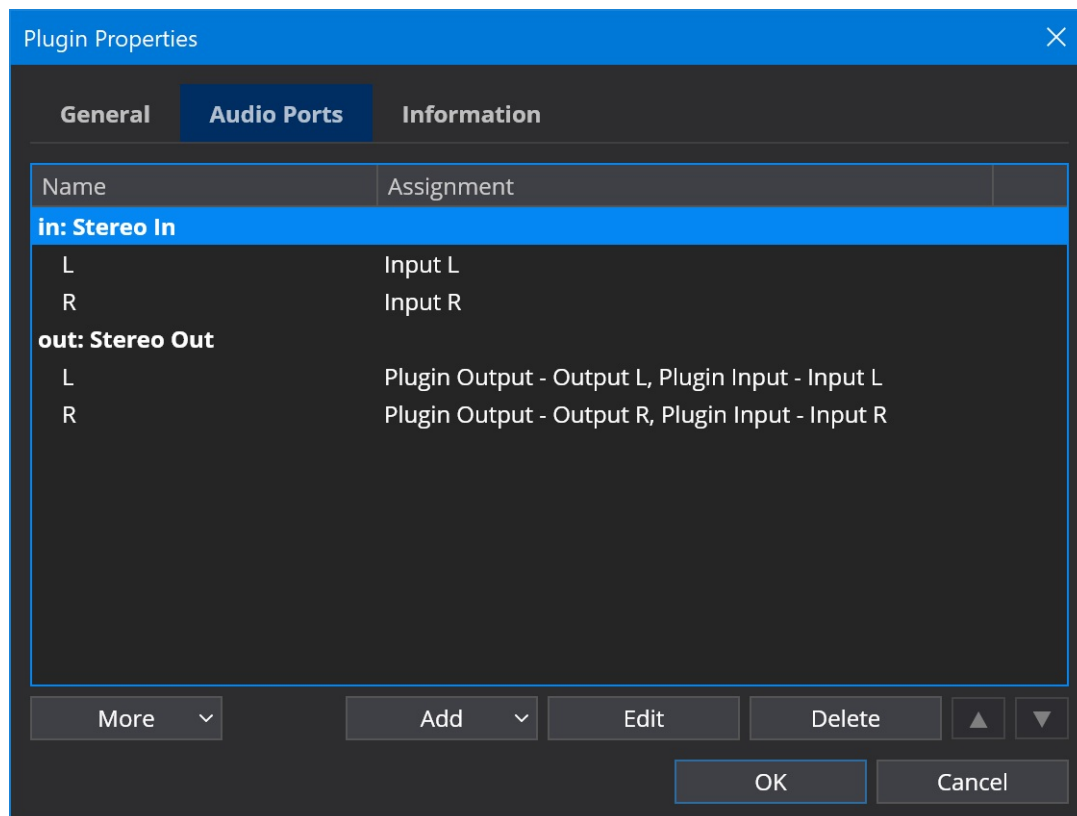
## Configuring Audio Ports

An audio port defines how a set of channels are mapped between an audio object and routes connected to the port. An audio port can be either an input port or an output port. For example:

- A plugin input port maps audio signals from connecting incoming routes to the plugin's input channels.
- A plugin output port maps audio signals from the plugin to connected outgoing routes.

An audio port consists of a set of channels and each channel has a number of assignments - each defining how an audio signal is mapped to or from the port.

The following screen shot shows the audio port settings for a typical audio effect plugin:



This plugin has a typical set of stereo in/stereo out audio ports. Note the two assignments on each of the output port channels which is explained below (see Wet/Dry Mix Control)

Notes on using this window:

- Click the Add button to add additional audio ports, or more channels to an existing port.
- Right clicking a channel allows for quickly creating simple 1 to 1 channel assignments.
- Double click a port or channel to edit its settings.

Editing a channel using the Edit Audio Channel window:

Assigned Channel	Mix Level
Plugin Output - Output L	100.0%
Plugin Input - Input L	100.0%

Notes:

- A channel can have multiple assignments to the target/source object
- Each assignment has its own mix level that controls the signal level of the assignment
- Depending on the context, each assignment may have a setting to control the pan and fade placement of this channel (see below).

## Audio Driver to Audio Port Mappings

Cantabile supports mapping of audio signals from the audio driver to a set of audio input/output ports. By defining audio ports in this manner, Cantabile's song files can be configured to work with a known set of audio ports and are isolated from the underlying hardware. Should the hardware configuration change, the audio ports can be reconfigured for the new hardware and the songs can continue to run unmodified.

Audio driver ports are defined in Options → Audio Ports and Cantabile maintains a separate set of assignments for each audio driver.

## Plugin Audio Port Mappings

The other place audio ports are used is on plugins. Each plugin instance has its own set of configurable audio ports. When a plugin is first inserted into a song Cantabile automatically creates a set of audio port configurations based on information from the plugin. Typically the default configuration will work fine however sometimes it might be necessary to change the audio port configuration. This is done by selecting the plugin in Cantabile's main window and choosing Edit → Plugin Audio Ports from the main menu.

## Wet/Dry Mix Control

In the screen shot above it was noted that the output port of the example plugin has two assignments for each output channel. If you take a closer look at these two assignments, you'll see the first is an assignment from the plugin's output channel (as expected), but there is also a second assignment from the plugin's input channel to the output port.

This second assignment is what allows wet/dry mix levels on effect plugins to work.

Audio output ports on plugins support creation of assignments from three distinct sources:

1. The plugin's output
2. The plugin's input
3. Any of the input port channels

The plugin's output is self explanatory and allows mapping the signal generated by the plugin to the output port. An assignment from the plugin's input sets up the dry signal to be used in wet/dry mixing. As the wet/dry mix level is adjusted for this plugin the output signal will cross fade from the plugin's input when dry to the plugin's output when wet.

Any assignments from an input port directly to an output port are pass-through audio signals and not affected by the plugin slot's mix or gain controls.

## Pan and Fade Settings

For plugin output ports each audio channel assignment has additional settings to control how it's affected by the plugin slot's pan and fade knobs and in each case allows setting a direction (none, left, right) or (none, front, back) and a pan or balance mode.

Balance mode provides typical stereo balance control, dropping the signal level on just one side towards zero, causing the sound to move in the other direction. Pan mode uses a panning law to place a mono signal - typically reducing the signal level when centered.

In simpler terms, balance mode should be used for a channel that is a component of a stereo or surround signal. Pan mode should be used for mono signals being placed in a stereo or surround environment.

## Mapping All Plugin Channels

By default Cantabile will automatically create one or two audio ports for both a plugin's input and outputs (depending on the functionality of the plugin). By only creating a limited set of ports it can reduce the processing load by not creating ports that won't be used.

Some plugins however might have additional audio channels that you might need to access. There are two ways you can map these channels to audio ports:

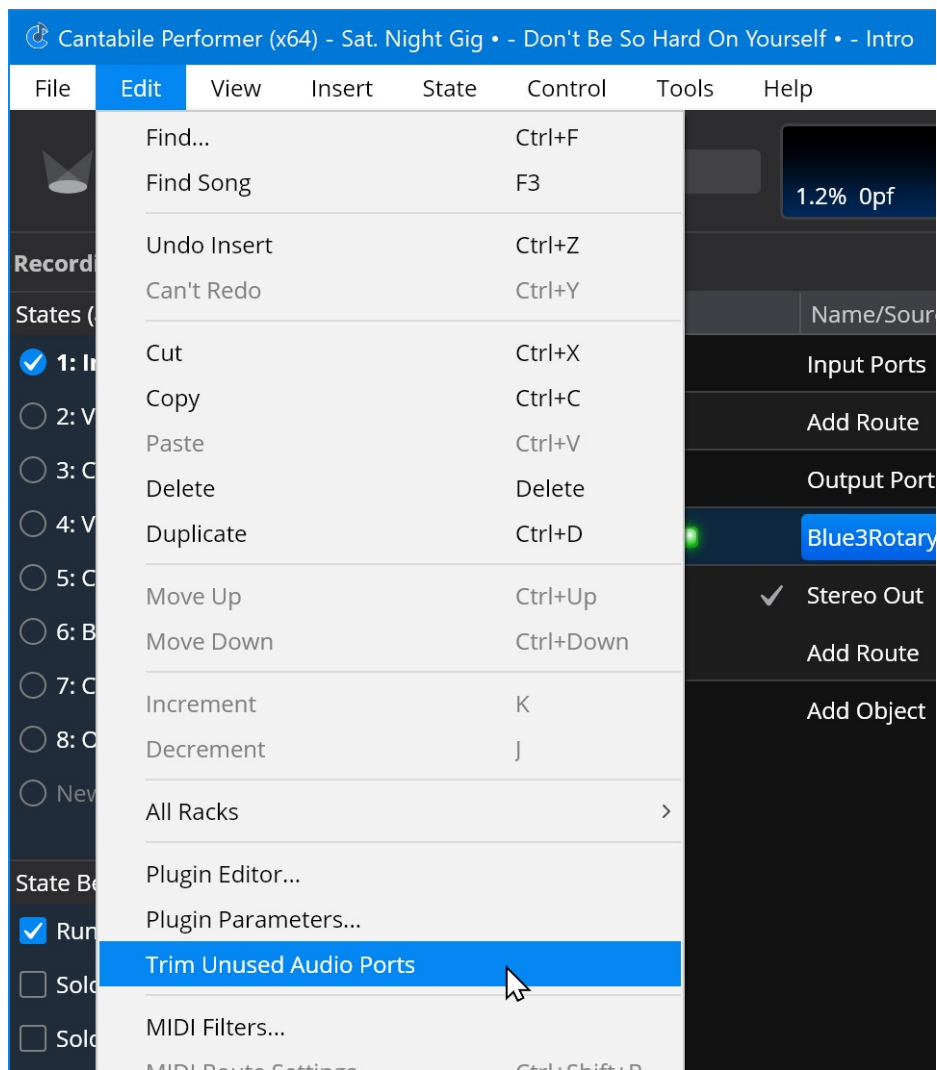
1. Manually create new ports and map them to the plugin's audio channels (as described above)
2. Right click on the plugin slot and choose "Map All Channels".

The second option will replace the existing audio ports with a new set that maps all of the plugin's channels. You can then either delete the the ports you don't actually need or use the Trim Unused Ports command (described below).

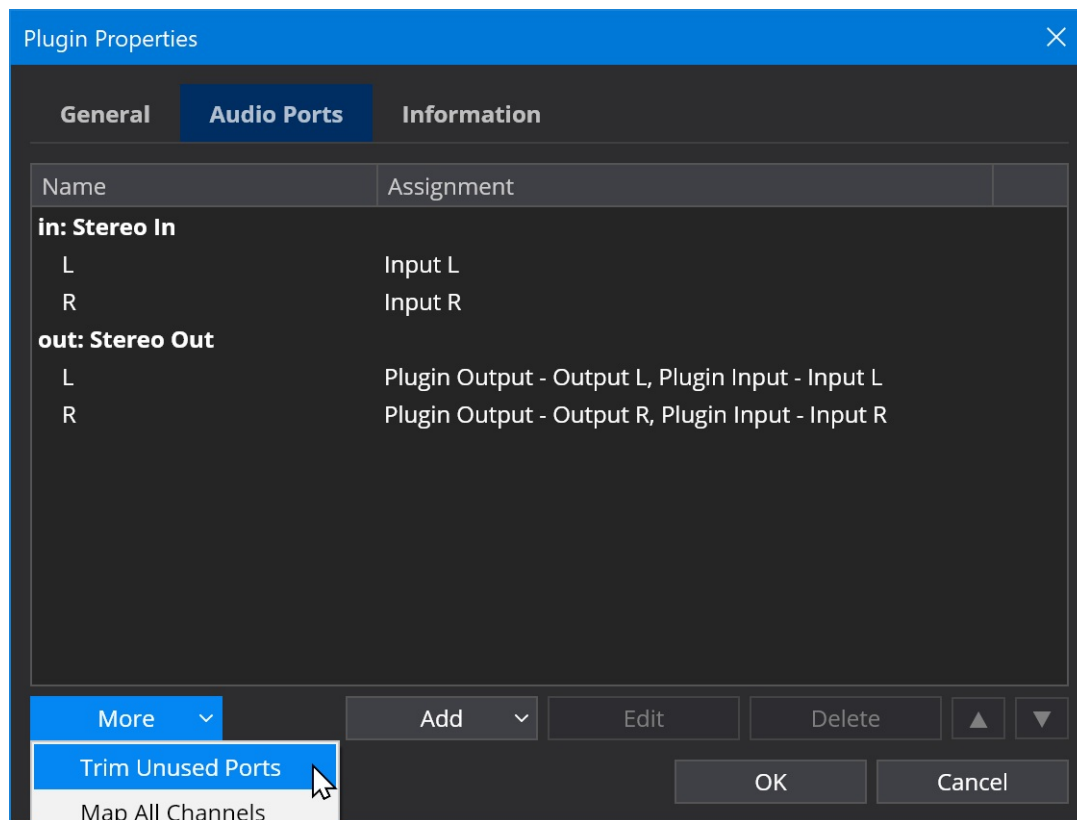
## Trimming Unused Plugin Ports

If you're using songs created from older versions of Cantabile where the default was to map all channels for new plugin instances, you might find those plugins have unused audio ports that aren't needed and may be contributing unnecessarily to the processing load.

To help mitigate against this you can use the "Trim Unused Ports" command to remove all ports for all plugins that aren't in use.



You can also trim the unused ports for a single plugin in its audio ports property page:



## Mapping Channels between Audio Ports

Cantabile supports connecting any audio output port to any audio input port. This raises the question of how two ports with a different number of channels in each are connected. Which channel connects to which?

The following rules describe how this is resolved:

- N to N - If the number of channels in the source audio port matches the number of channels in the target audio port, the channels are mapped 1 to 1.
- 1 to N - If the source audio port has one channel and the target audio port has multiple channels, the one source channel is mapped to every target channel.
- N to 1 - If the source audio port has more than one channel and the target audio port has one channel, all the source channels are mapped to the one target channel, with the mix level divided down by the number of source channels. eg: mapping a stereo port to a mono port creates two assignments each at 50% mix level.
- N to M - If the number of channels in the source and target don't match and they both have more than 1 channel, Cantabile will create one to one assignments for as many channels as it can and leave the others disconnected.

Note that you can't change the above assignment rules - Cantabile always connects audio ports in this way. If two audio ports are incompatible and you need to connect the two objects, [create another compatible audio port](#) on either object and connect using that port.

## MIDI Route Settings

MIDI routes provide connections between MIDI ports and provide a number of controls that determines which MIDI events are passed through and how those events are manipulated before being forwarded to their destination.

MIDI route from Main Keyboard

Channel

Source Channel(s):

None
Omni
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Target Channel:

Same
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Key Range

Lowest Note:

0 - C-2

Highest Note:

127 - G8

Transpose:

0.0 = perfect unison

☐ Learn Mode

☐ Ignore Global Transpose Settings

☐ Notes Only (suppress all other events)

☐ Show range on keyboard as:

(Target Color)

Velocity Curve

Min. Input:

0

Max Input:

127

Min. Output:

0

Max Output:

127

☒ Expand/Compress
☐ Enhance

Program

☐ Only route when this program number is active:

1

Condition

☐ Only route when CC:

0

is greater or equal to

64

Reset

OK

Cancel

#### Source channel

Only routes events matching one or more MIDI channels

#### Target channel

Changes routed MIDI events to the selected MIDI channel

#### Note Range (Lowest Note/Highest Note)

Limit events to a particular keyboard note range (ie: keyboard splits)

#### Transpose

Transpose all notes by the specified interval

#### Learn Mode

Allows learning the note range and transpose settings (see below)

#### Ignore Global Transpose Setting

When selected the global transpose setting is ignored for this route. Useful when routing to rhythmic instruments (drum kits etc...) *Cantabile Performer Only.*

#### Notes Only (Suppress All Other Events)

Only routes note and after touch events. All other events (eg: controllers) are suppressed. Useful when you have two separate note ranges mapped using two routes to the one instrument. By selecting this setting on one route the target



instrument will only get one set of controller events. This option does not suppress the MIDI CC 123 event (All Notes Off).

Show On Keyboard As: (choose color)

When selected the keyboard range is shown on Cantabile's on-screen keyboard. You can also select a color.

Velocity Curve

Adjusts the velocity curve for MIDI note and after touch events.

Only Route When This Program Number is Active

Restricts routing to only take place if the specified program number was the last received program event.

Condition

Restricts routing to when a particular CC condition is true. eg: only route when a particular pedal is pressed.

## MIDI Filters

Every MIDI route also supports a set of MIDI Filters that provide even more fine grained control over MIDI passing through the route. To edit the MIDI Filters for a route, select the route in Cantabile's main window and choose Edit → MIDI Filters from the main menu. Note that MIDI Filters are applied after the regular MIDI route settings shown above.

## Learn Mode

You can simplify the entry of key range and transpose settings by using Learn Mode:

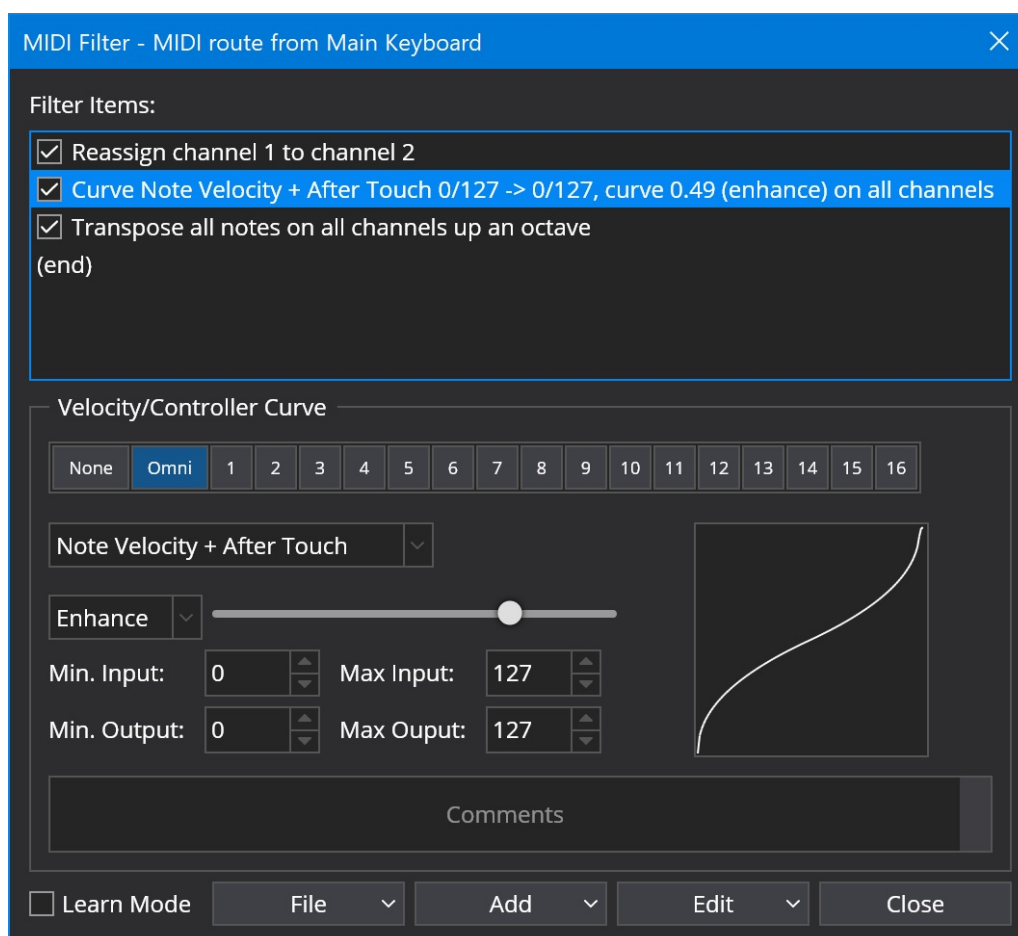
1. Turn on the "Learn Mode" option
2. Click to place focus in either of the note range fields, or the transpose field
3. Play a note to set the field's value (the note must be played on the same device as the route is connected to).
4. Turn off learn mode

To set the transpose, play two notes - the interval will be calculated based on the previous two notes played. eg: playing C4 then E4 will set the transpose to "+0.4 - up a major 3rd".

## MIDI Filters

*Cantabile Solo and Cantabile Performer Only.*

MIDI Filters support applying a wide range of transformations to MIDI events as they pass through the system.



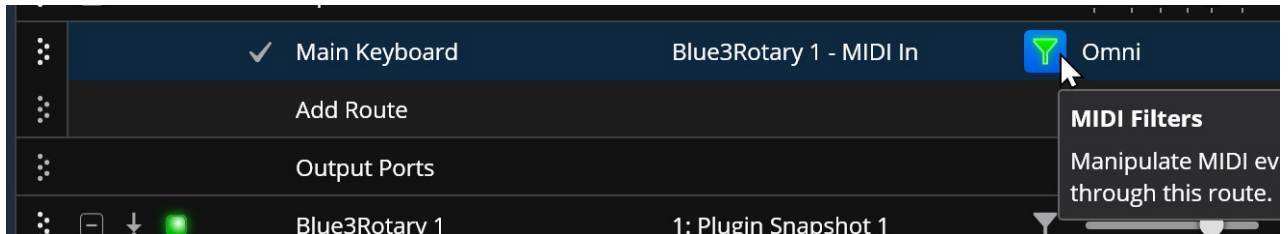
## Accessing MIDI Filters

MIDI Filters are available at many different points:

- MIDI Routes - right click on the route and choose MIDI Filters
- Plugins - right click on the plugin and choose MIDI Filters

Each MIDI filter applies to that point in the MIDI processing pipeline.

You can also access the MIDI filters via the MIDI filter button in the object's slot in the main window:



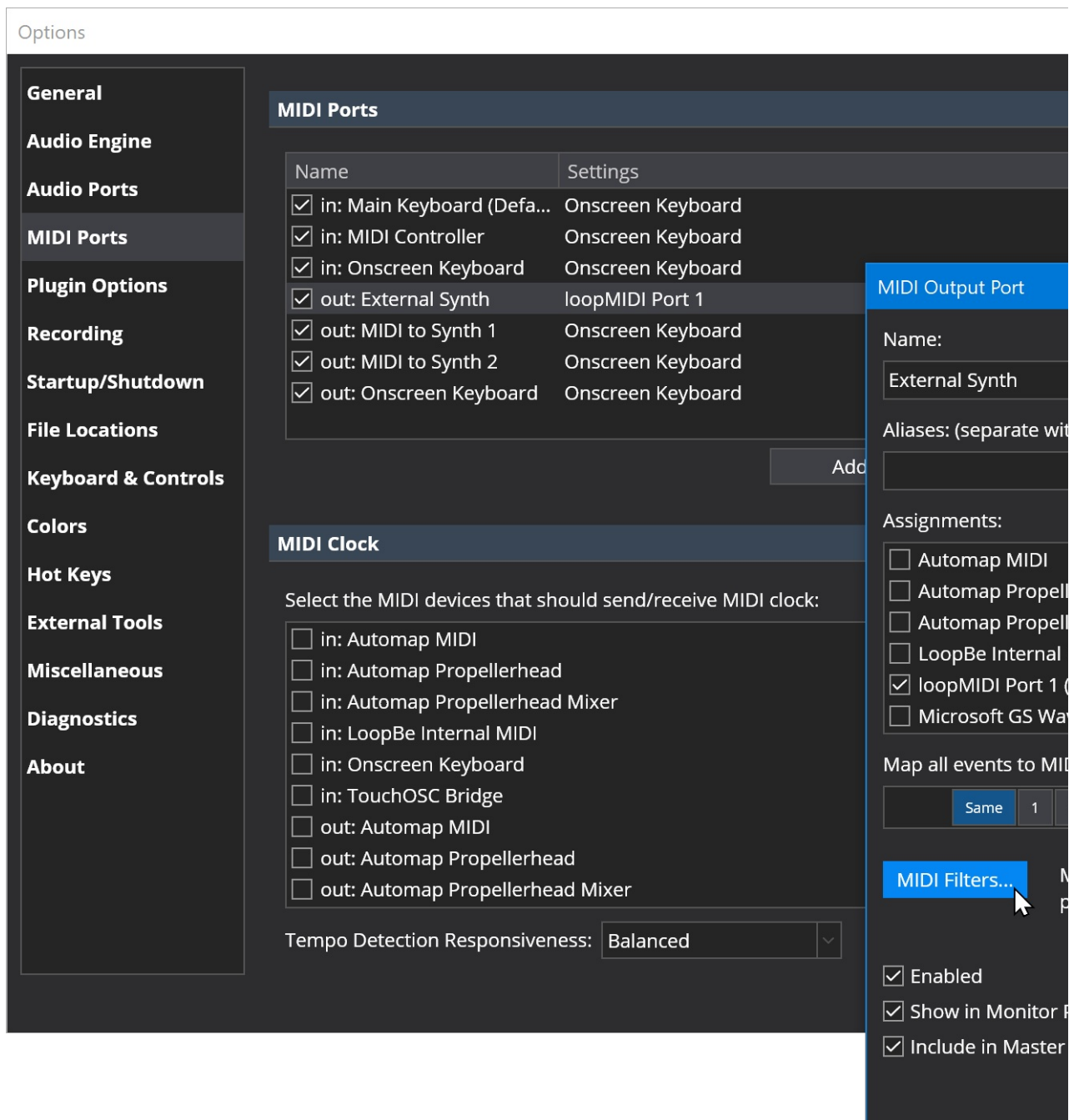
Note the button lights up green indicator to indicate there are active MIDI Filters on this object.

## Global Port MIDI Filters

Global MIDI ports also have MIDI Filter support and provide a convenient way to massage incoming MIDI events across all songs.

For example you can add MIDI Filters to an environment MIDI Input port.

1. Open Cantabile's Options
2. Switch to the MIDI Ports page
3. Double click on a MIDI Port
4. Click the MIDI Filters button



## Object MIDI Port Files

Media Player and Racks also support MIDI filters on the MIDI Ports:

- Media Players - right click the media player, choose Properties, switch to the MIDI Ports tab, double click the port and choose MIDI Filters
- Rack Ports - open the rack and from the File menu choose Rack Properties, switch to the MIDI Ports tab, double click the port and choose MIDI Filters

## MIDI Filter Processing Order

- For MIDI Routes, MIDI Filters are applied after the other MIDI route settings.
- For Plugins, MIDI Filters are applied before the events are forwarded to the plugin.

## MIDI Filter Items

Cantabile supports a number of different filter items which are explained below.

A couple of settings are common:

Channels

It is possible to indicate the source and target channel(s) for the events. It is possible to select multiple channels by holding the control key while clicking the channel numbers.

#### Copy event

Indicate whether to suppress or copy the original event.

#### Edge Triggered

If checked the source controller is treated as a momentary MIDI button that sends a value > 64 when pressed and < 64 when released. This kind of binding is edge-triggered in that it invokes the target when the value crosses from less than 64 to greater than 64. If unchecked the source controller is expected to send a non-zero value when pressed and no event or a CC value of 0 when released. ie: it triggers on any non-zero value.

#### Out of range

Choose how to handle event values that are outside a specified range:

##### Discard

values outside the source range are discarded and the binding isn't invoked.

##### Clamp

values are clamped to the source range and the binding invoked.

##### Slide

the source range is internally adjusted up/down to make the source value within range.

##### Passthrough

the value is passed through the filter item unchanged.

## Aftertouch To Controller Map

After Touch to Controller Map

Apply to Channel:

None	Omni	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
------	------	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Source: Average of all held notes Note: 60 - C3

Target Controller: 0 Channel: Same

The After Touch To Controller Map item generates a controller event based on aftertouch values. The aftertouch value can either be from a single note or calculated as the average or maximum of all held notes. This calculated value is sent as the controller event's value.

#### Apply to Channels

The MIDI channels this filter should be applied to.

#### Source

How to calculate the after touch value

- Maximum of All Held Notes
- Average of All Held Notes
- After touch of a specific note

#### Note

The note to use when "After touch of a specific note" is selected.

#### Target Controller

The controller number of the generated controller events

#### Channel

The channel number of the generated controller events

## Button Bank

**Button Bank**

Apply to Channel:

None	Omni	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
------	------	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bank A Controllers: 21-24 Base 0 Multiplier 10

Bank B Controllers: 30-39 Base 0 Multiplier 1

Send Event: Controller Controller 100

The Button Bank item allows mapping two ranges of button controllers onto some other MIDI event.

Say you have two banks of button controllers that you'd like to translate to say a program change event. The button bank controller allows this by combining the button presses from one bank with the button presses from the other (using configurable add/multipliers operations) then sends the desired event.

Say you have two banks of button controllers that you'd like to translate to say a program change event. The button bank controller allows this by combining the button presses from one bank with the button presses from the other (using configurable add/multipliers operations) then sends the desired event.

It supports the following settings:

#### Apply to Channels

The MIDI channels this filter should be applied to.

#### Bank A Controllers

The range of button controllers for bank A

#### Bank A Base

Base value for bank A

#### Bank A Multiplier

Multiplier for bank A

#### Bank B Controllers

The range of button controllers for bank B

#### Bank B Base

Base value for bank B

#### Bank B Multiplier

Multiplier for bank B

#### Send Event

The event to send. Choose from

- Bank Select + Program Change,
- Program Change
- Bank Select
- Controller
- RPN Registered Parameter (Coarse)
- RPN Registered Parameter (Fine)
- NRPN Non-registered Parameter (Coarse)
- NRPN Non-registered Parameter (Fine)

#### Controller

The controller or RPN/NRPN parameter number to send.

For example, assume you have 20 MIDI push buttons - the first set of 10 on MIDI controllers 80-89 and the second set on controllers 90-99. When you press a button in bank A, the filter remembers which one you pressed. When you press a button in bank B, the filter generates a program change event. The program number it generates is calculated by this formula:

$$(\text{IndexOfButtonA} + \text{BankABase}) * \text{BankAMultiplier} + (\text{IndexOfButtonB} + \text{BankBBase}) * \text{BankBMultiplier}$$

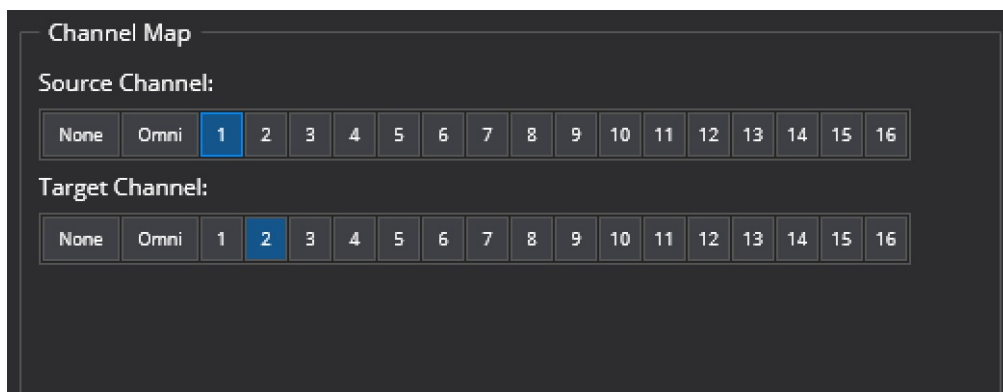
So say you pressed button 81 then button 93 that would be:

$$(1 + 0) * 10 + (3 + 0) * 1 = 13$$

In other words, you're using bank A as the tens column, bank B as the units and you've got a really quick way to access 100 program changes from 20 buttons.

Other notes: \* Each bank is optional so you can use it for a single button bank \* If the controller range for a bank only has a single controller, it uses the controller value rather than its index. This allows other manipulations of controllers to be created. \* The button controllers don't need to be in order. eg: 10-14,20,8 is a valid definition for 7 buttons.

## Channel Map



Channel Map

Source Channel:

None	Omni	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
------	------	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Target Channel:

None	Omni	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
------	------	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

The Channel Map filter item assigns all MIDI events from one or more MIDI channels to one or more other channels.

### Source Channels

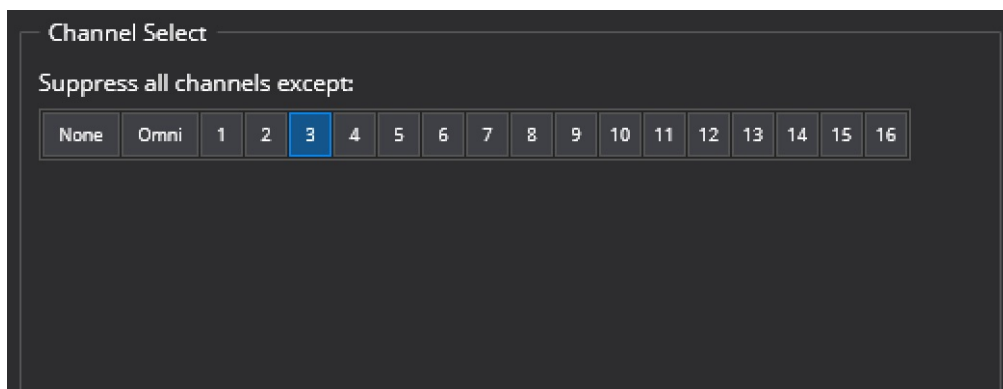
The channel or channels to be remapped.

### Target Channels

The new channel or channels.

Note: you can select multiple channels by holding the Control key while clicking the channel numbers.

## Channel Select



Channel Select

Suppress all channels except:

None	Omni	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
------	------	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

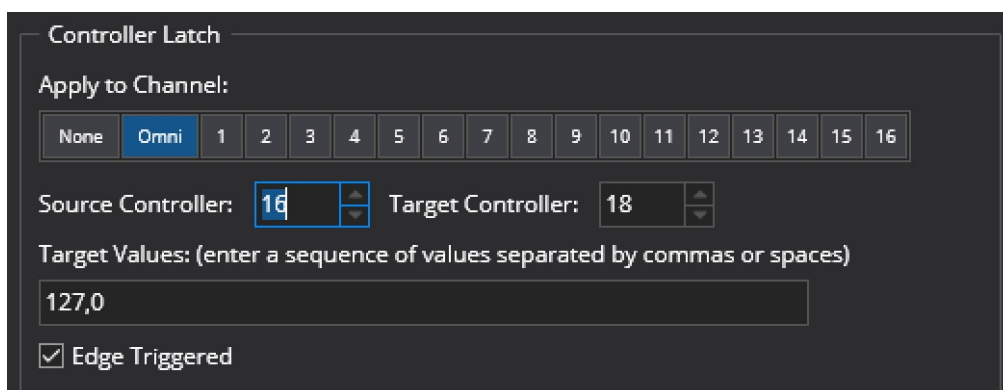
The channel select item filters out all MIDI events except those on one (or more) channels.

### Source Channel(s)

The channel or channels to be selected

Note: you can select multiple source channels by holding the Control key while clicking the channel numbers.

## Controller Latch



Controller Latch

Apply to Channel:

None	Omni	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
------	------	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Source Controller: 16 Target Controller: 16

Target Values: (enter a sequence of values separated by commas or spaces)

127,0

☒ Edge Triggered

Transforms an incoming MIDI controller button or pedal press to a sequencer of controller values.

### Apply to Channels

The MIDI channels this filter should be applied to.

### Source Controller

The controller number of the source button/pedal.

### Target Controller

The controller number of events generated by this filter. Can be the same as the source controller.

### Target Values

A sequence of controller values. On each press of the source controller button/pedal the next value in the sequencer will be generated as a new controller event.

This filter is intended for cycling through VST parameters that have discrete values rather than continuous controller values.  
Eg: this could be used to assign a push button to toggle a VST parameter on/off.

## Controller Map

The screenshot shows the 'Controller Map' settings window. At the top, 'Apply to Channels:' has a row of buttons from 'None' to '16', with '1' selected. Below this, there are three columns: 'Controller Kind', 'Controller', and 'Range'. The 'From:' row shows 'CC Coarse' for both 'Controller Kind' and 'Controller', with a range from '30' to '100'. The 'To:' row shows 'CC Coarse' for 'Controller Kind', 'Same' for 'Controller', and a range from '0' to '127'. At the bottom, 'New Channel:' is set to 'Same', there is an unchecked 'Copy Event' checkbox, and 'Out of range:' is set to 'Slide'.

The Controller Map filter item maps all MIDI events for one MIDI controller to another MIDI controller. It is possible to map between nearly all kind of events, and to specify a value range to use for both source and target, thereby also scale the values of the events as needed.

### Apply to Channels

The MIDI channels this filter should be applied to

### From Controller Kind

The source controller kind to be mapped

### From Controller

The source controller number to be mapped

### From Range

The range of the source controller to be mapped

### To Controller Kind

The kind of controller event to map to

### To Controller

The controller number to map to

### To Range

The target range of the mapped events

### New Channel

Optional remaps the events to a different MIDI channel

### Copy Event

When selected, the original event is also passed through unmodified

### Out of Range

How to handle out of range source values ([see above](#)).

## Controller To Note

The screenshot shows the 'Controller to Note' settings window. 'Apply to Channel:' has buttons from 'None' to '16', with '1' selected. Below, 'Controllers:' is a text box containing '16-21'. 'Notes:' is a text box containing 'C3-F3'. 'Velocity:' is set to 'Use CC Value'. 'Release Mode:' is set to 'When CC Value 0'. At the bottom, the 'Suppress Original Event' checkbox is checked.

Maps controller events to notes.

### Apply to Channels

The MIDI channels this filter should be applied to

### Controllers

The set of MIDI controller numbers that should be mapped

## Notes

The set of MIDI notes that should be generated where each note maps to the corresponding controller number in the same position. The number of notes must match the number of controllers.

## Velocity

The velocity value of the generated notes. Can be based on the value of the mapped controller event, or a specific velocity value

## Release Mode

When the generated note should be released

- When CC Value 0
- Immediately
- Never

## Suppress Original Event

When selected the original event is not passed through

## Controller To Program Map

Controller to Program Map

Apply to Channel:

None Omni 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Select Program CC: 16

Next Program CC: 17 Previous Program CC: 18

New Channel: Same

☐ Duplicate Event

Maps a MIDI controller event to a MIDI program change event and/or maps controller button presses to next/previous program.

## Apply to Channels

The MIDI channels this filter should be applied to.

## Select Program CC

The controller number to be mapped directly to a MIDI program change. The controller's value becomes the program number sent.

## Next Program CC

The controller number of a button to select the next program.

eg: if controller 63 is mapped to the next program, and the last program change event was 42, then pressing controller 63 will yield a program change 43 event.

## Previous Program CC

The controller number of a button to select the previous program.

## New Channel

The new channel number for the generated program change events.

## Duplicate Event:

When cleared, incoming matching controller events are suppressed and new program change events generated. When set, the incoming controller events are passed through unaltered while still generating program change events.

## Controller To Switch

Controller to Switch

Apply to Channels:

None Omni 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

From: CC Coarse 16 Threshold: 64

To: CC Coarse 21 Off/On: 0 127

New Channel: Same ☐ Copy Event

Maps a continuous controller event to an on/off controller event.



From Controller Kind

The kind of MIDI event to convert from

From Controller Number

The controller number of events to be converted

Threshold

The minimum value of the controller that switches the target controller on

To Controller Kind

The kind of MIDI event to generate

To Controller

The controller number of generated events

Off

The controller value to send when the source controller is less than the threshold

On

The controller value to send when the source controller is greater or equal to the threshold

New Channel

The new channel number for the generated program change events.

Copy Event

When cleared, incoming matching controller events are suppressed and new events generated. When set, the incoming controller events are passed through unaltered while still generating new events.

## HUI Switch Decoder

HUI Switch Decoder

Apply to Channel:

None 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Target Base Channel:

None 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Zone 0 ports will be mapped to CC 0-7, Zone 1 port will be mapped to CC 8-15 etc...

Zones 16 and above will be mapped to the next channel after the target base channel

The HUI spec encodes button presses using two CC messages and uses a concept of zones and ports. A zone typically represents a group of buttons such as a channel strip while a port represents a particular button in that zone. Each zone has 8 ports and there are about 30 defined zones.

The HUI Switch Decoder filter map each button to a CC as follows:

$CCNumber = zoneNumber * 8 + portNumber$

When pressed value of 127 is sent, when released 0. ie: a standard MIDI button style behaviour. For example: zone 14 port 4 would be mapped to  $14 * 8 + 4 = CC \#116$ .

As there aren't enough CC numbers to cover the full range of zones, if the calculated CC number exceeds 127 then the CC numbers wrap around and the next channel number is used.

A few more details are available in [this blog post](#)

## Key Range

Key Range

Apply to Channel:

None Omni 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Key Range: 36 - C1 To: 60 - C3

Transpose: +2.0 = 2 octaves Channel: Same

☐ Notes only ☒ Suppress out of range notes ☐ Passthrough original

The Key Range filter item transposes a range of notes up or down by a specified interval. It is possible to change the channel of the notes, and indicate if notes outside the range should be suppressed. Also, if the target channel is different from the source channel then it is possible to indicate whether or not to map other events than notes to the new channel.

The key range filter is similar to the keyboard split filter except instead of creating a single split above/below one point on the keyboard it allows an arbitrary range on the keyboard to be selected.

The key range filter can be used to either remap a range of the keyboard to a new MIDI channel, or to simply suppress notes outside the specified range.

#### Apply to Channels

The MIDI channels this filter should be applied to

#### Key Range

Specifies the first and last note in the range

#### Channel

Specifies a new channel number for all notes matching the specified keyboard range

#### Transpose

The interval to transpose notes in the keyboard range by

#### Notes Only

When selected, only remaps note events. When cleared, all other events are also remapped.

#### Suppress out of range notes

When selected all notes outside the specified range are suppressed. When cleared, notes outside the specified range are passed through the filter unaltered.

#### Passthrough Original Events

When cleared, notes that are remapped are modified and subsequent filter items only see the modified event. When selected, notes that are remapped generate a new event and the original event is passed to the next filter item unaltered.

Also consider the related filter types [Keyboard Split](#) and [Transpose](#)

## Key Range To Note

The screenshot shows the 'Key Range to Note' filter configuration. At the top, the title 'Key Range to Note' is displayed. Below it, the 'Apply to Channel:' section features a row of buttons from 'None' to '16', with 'Omni' selected. The 'Key Range:' is set to '60 - C3' and the 'To:' is set to '84 - C5'. The 'Target Note:' is set to '60 - C3'. At the bottom, there are two checkboxes: 'Retriggerable' (unchecked) and 'Suppress out of range notes' (checked).

Maps a range of notes to a single note on the same channel, and keeps the velocity.

#### Apply to Channels

The MIDI channels this filter should be applied to

#### Key Range

The range of notes to be mapped

#### Target Note

The target note to be generated when any note in the range is played

#### Retriggerable

When set, pressing a second note while still holding the first will trigger a second target note. When clear, all notes must be released before the target note can be triggered again.

#### Suppress Out of Range Notes

When set, any notes outside the key range will be suppressed

## Keyboard Split

The screenshot shows the 'Keyboard Split' filter configuration. At the top, the title 'Keyboard Split' is displayed. Below it, the 'Apply to Channel:' section features a row of buttons from 'None' to '16', with 'Omni' selected. The 'Split Point:' is set to '60 - C3' with a note '(first note in upper range)'. The 'Lower channel:' is set to '1' and the 'Upper channel:' is set to '2'. The 'Transpose:' for the lower channel is set to '+1.0 = 1 octave' and for the upper channel is set to '-1.0 = 1 octave'. At the bottom, there is a checkbox for 'Notes only' which is unchecked.

Splits a keyboard range by a specified note, and maps the lower and upper part to two channels, potentially also transposed. It is possible to choose to have other events than notes mapped to the new channels as well.

The keyboard split item splits incoming MIDI events into two separate MIDI channels.

#### Apply to Channels

The MIDI channels this filter should be applied to

#### Split Point

The first note of the upper keyboard range

#### Lower Channel

The new channel for notes below the split point

#### Upper Channel

The new channel for notes above the split point

#### Lower Transpose

The interval to transpose notes in the lower range by

#### Upper Transpose

The interval to transpose notes in the upper range by

#### Notes Only

When selected, only MIDI note events are routed. When cleared (the default), MIDI controllers and other events are duplicated and sent to both target channels.

Also consider the related filter types [Key Range](#) and [Transpose](#).

## Note As Controller

The screenshot shows the 'Note as Controller' settings window. At the top, 'Apply to Channel:' is set to 'Omni'. Below this is a row of buttons for channels 1 through 16. The 'Mode:' dropdown is set to 'Single variable controller'. The 'Condition:' dropdown is set to 'None'. The 'Key range:' is set from '48 - C2' to '72 - C4'. The 'Min value:' is set to '0'. The 'Target CC:' is set to '0'. The 'Channel:' dropdown is set to 'Same'. The 'Max value:' is set to '127'.

Maps a range of notes to either a single controller, or a range of controllers. It is possible to indicate a condition, ie a controller where the value must be >63, otherwise the filter item will be bypassed.

If the range of notes maps to a single controller, then the index of the source note within the range of notes is used to calculate the controller value.

If the range of notes maps to a range of controllers, then the index of the source note within the range of notes is used to calculate the controller number, and the value for the controller is as indicated by the settings for Released and Pressed.

The Note as Controller filter item emulates a MIDI controller using note events. This filter can emulate either a single variable controller - where a range of notes represents the value of the controller, or a set of button controllers - where each note represents one controller in a series of on/off controllers.

It supports the following settings:

#### Apply to Channels

The MIDI channels this filter should be applied to.

#### Mode

Whether to emulate a single variable controller or a range of button controllers.

#### Condition

The number of a controller whose value must be pressed ( $\geq 64$ ) for the emulation to take effect. (see explanation below).

#### Key Range

The range of notes used to emulate the controller or controllers.

#### Target CC

The controller number to emulate.

#### Channel

The channel to send controller events on (or Same to use the same channel as the source note events).

#### Min Value (when emulating a variable controller)

The minimum value of the emulated controller - mapped to the lowest note in the key range

#### Max Value (when emulating a variable controller)

The maximum value of the emulated controller - mapped to the highest note in the key range.

#### Released (when emulating button controllers)

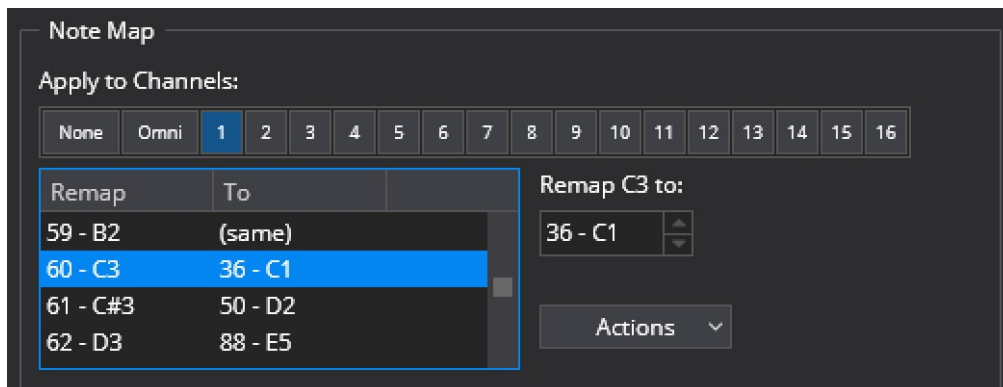
The value of the controller when the note is released - typically 0.

#### Pressed (when emulating button controllers)

The value of the controller when the note is pressed - typically 127.

The condition controller can be used to enable/disable controller emulation using another controller. For example you might define a range of notes to act as a series on/off buttons but only want them to take effect when a real MIDI controller button is pressed, or a pedal maybe. To do this, set the Condition property to the controller number of the button/pedal that enables the emulation. The condition controller must take on a value of 64 or greater when pressed for this feature to work.

## Note Map



Provides a one-to-one mapping table for every note.

### Apply to Channels

The MIDI channels this filter should be applied to

### Note List

A list of all notes showing the current remapping

### Remap XX to

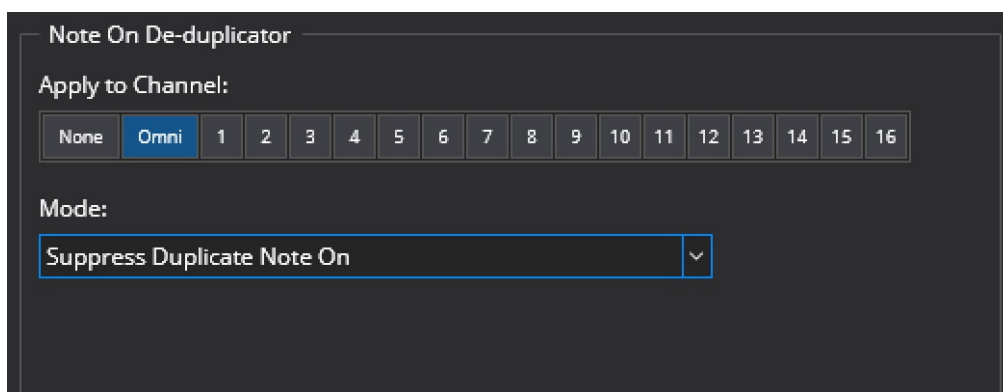
Sets the target note for the note selected in the list

### Actions

Additional commands for working with the Note Map filter:

- Load One to One Map - loads a map where all notes are mapped to themselves
- Load Suppress All Map - loads a map where all notes are suppressed
- Copy Map - copies the current map to the clipboard
- Paste Map - pastes the clipboard to the current map
- Import - imports a note map from a file
- Export - saves the note map to a file

## Note On De-duplicator



Suppresses repeated note-on events for the same note that have no intervening note off event.

### Apply to Channels

The MIDI channels this filter should be applied to

### Mode

How duplicated notes should be handled

- Suppress Duplicate Note On
- Insert a Note Off before the duplicate Note On

This filter can be used to suppress repeated note on events from "triple sensor" and other MIDI keyboards that simulate the action of a grand piano - where repeated notes can be played without engaging the dampers.

For plugins and synths that expect exactly one note off event for every note on event, these keyboards can cause stuck notes. This MIDI filter provides a way to use these keyboards with these synths.

## Note To Program

The screenshot shows the 'Note to Program' MIDI filter interface. It has a title bar 'Note to Program'. Below it is a section 'Apply to Channel:' with a row of buttons: 'None', 'Omni', and numbered buttons 1 through 16. 'Omni' is selected. Below this is 'Note range:' with two input fields: '36 - C1' and '60 - C3', separated by 'to'. Below that is 'Note Kind:' with a dropdown menu showing 'White Notes'. At the bottom is 'Base Program Number:' with an input field showing '1' and a checkbox labeled 'Banked Program Change' which is unchecked.

Maps note events to program change events

Apply to Channels

The MIDI channels this filter should be applied to

Note Range

The range of notes to be mapped to program changes

Note Kind:

Which notes should be mapped

- All Notes
- White Notes
- Black Notes

Base Program Number

The program number of the first note in the range

Banked Program Change

Whether to send a plain program change or a banked program change

See [this video](#) for a walkthrough of this filter item.

## Parameter Automation

The screenshot shows the 'Parameter Automation' MIDI filter interface. It has a title bar 'Parameter Automation'. Below it is a section 'Apply to Channel:' with a row of buttons: 'None', 'Omni', and numbered buttons 1 through 16. 'Omni' is selected. Below this is 'Source Controller:' with an input field showing '16'. To its right is 'Range' with two input fields: '0' and '127', separated by 'to'. Below that is 'Target Parameter:' with a dropdown menu showing 'Drive'. To its right is 'Range' with two input fields: '0.00' and '1.00', separated by 'To'.

The Parameter Automation filter item assigns a MIDI controller to a parameter of a VST plugin.

Apply to Channels

The MIDI channels this filter should be applied to

Source Controller

The MIDI controller to be used to control the parameter

Range (Source)

Range of values for the source controller. Values outside this range will be clamped to the specified range

Target Parameter

The parameter of the VST plugin to be manipulated by the MIDI controller

Range (Target)

Range of parameter values for the target parameter

Notes:

- This filter item is provided primarily for backwards compatibility and [Bindings](#) provide a more flexible mechanism for mapping MIDI controllers to settings in Cantabile.

- This filter is only available when editing the MIDI filters of a VST plugin.

## Pitch Based Velocity Ramp

**Pitched Velocity Ramp**

Apply to Channel:

None Omni 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Key Range: From: 60 - C3 To: 84 - C5

Adjust velocity by: From: 80.00 To: 90.00

Adjust notes: ☒ Below ☒ Above

The Pitch Based Velocity Ramp adjusts the velocity of notes according to their pitch. This filter is useful for adjusting instruments that get too loud or too quite at different parts of the keyboard.

It is possible to indicate the key range to be affected, and how much the velocity should be adjusted across the key range. It is also possible to indicate if notes below and above the key range should be adjusted or left at their original velocity. These options are useful if you want to create a sequence of pitched velocity ramps for different key ranges.

### Apply to Channels

The MIDI channels this filter should be applied to

### Key Range

Specifies the first and last note of the range of notes whose velocities are to be adjusted

### Adjust Velocity By

Specifies a percentage to adjust the velocity of notes by. The first value specifies the velocity adjustment at the low end of the key range. The second value specifies the velocity adjustment at the high end of the key range. Notes within the range will be adjusted by a percentage calculated by linearly interpolating these two values.

### Adjust Notes Below

Specifies if notes below the specified key range should be adjusted.

### Adjust Notes Above

Specifies if notes above the specified key range should be adjusted.

The Adjust Notes Above/Below options are useful if you want to create a sequence of pitched velocity ramps for different key ranges.

## Program Map

**Program Map**

Apply to Channel:

None Omni 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Source Program: 65

Target Program: 48

New Channel: Same

Maps a single program change command to another program change command. Supports banked program changes.

### Apply to Channels

The MIDI channels this filter should be applied to

### Source Program

The program number to remap from

### Target Program

The program number to remap to

### New Channel

The channel number of the generated program change event

## Sostenuto

The image shows the 'Sostenuto' MIDI filter settings. At the top, the title 'Sostenuto' is followed by a horizontal line. Below this, the label 'Channels:' is positioned above a row of 17 buttons. The buttons are labeled 'None', 'Omni', and then numbers 1 through 16. The 'Omni' button is highlighted with a blue border. The rest of the interface is a solid dark grey.

The Sostenuto midi filter implements the sostenuto pedal from a grand piano, it's similar to the damper/sustain pedal except it only holds the notes that were on at the time the pedal was pressed.

Apply to Channels

The MIDI channels this filter should be applied to

The MIDI controller for Sostenuto is CC#66.

## Sticky Note

The image shows the 'Sticky Note' MIDI filter settings. The title 'Sticky Note' is followed by a horizontal line. Below it, the label 'Sticky Note:' is above a row of 17 buttons: 'None', 'Omni', and numbers 1-16. The 'Omni' button is highlighted with a blue border. Below the buttons are four settings: 'Max Polyphony:' with a numeric input set to '3' and a dropdown arrow, followed by '(0 for unlimited)'; 'Master Release Note:' with a text input '36 - C1' and a dropdown arrow; 'Master Release CC:' with a text input 'None' and a dropdown arrow; and a checkbox labeled 'Retriggerable' which is currently unchecked.

Keeps a note playing until specifically released, either by the specified Master Release Note or the specified Master Release Controller. It can either be monophonic, in which case the previous note is released when a new note is played, or polyphonic, then all notes are kept playing until released.

Apply to Channels

The MIDI channels this filter should be applied to

Max Polyphony

The maximum number of notes to be held at one time. When the limit is reached the oldest held note is released and the new note held.

Master Release Note

On receiving this note, all currently held notes will be released

Master Release CC

On receiving the controller, all currently held notes will be released

Retriggerable

Whether the same note can be retriggered a second time while still held

## Suppress Events

The image shows the 'Suppress Events' MIDI filter settings. The title 'Suppress Events' is followed by a horizontal line. Below it, the label 'Apply to Channel:' is above a row of 17 buttons: 'None', 'Omni', and numbers 1-16. The 'Omni' button is highlighted with a blue border. Below the buttons are two settings: 'Suppress Notes & AT:' with a dropdown arrow, a text input 'C5-C6', and an example 'eg: A0,C2-C4'; and 'Suppress Controllers:' with a dropdown arrow, a text input '(none)', and an example 'eg: 1,2,3-30,45'. At the bottom, there are six checkboxes: 'All Notes' (unchecked), 'Program Change' (checked), 'Pitch Bend' (unchecked), 'All After Touch' (unchecked), 'Channel Pressure' (unchecked), and 'MIDI Clock' (unchecked).

Filters out notes, controllers, after touch, program change, channel pressure, pitch bend or midi clock on one or more channels. It is possible to indicate a range for notes and for controllers, and also choose whether the range is included or excluded from filtering. The range can be empty, which makes it possible to indicate an empty range of allowed notes, the same as suppressing all notes.

#### Apply to Channels

The MIDI channels this filter should be applied to

Notes and Aftertouch (first row of settings)

Which notes and after touch events to suppress or allow

Controllers (second row of settings)

Which controllers to suppress or allow

All Notes

Suppresses all notes

All After Touch

Suppresses all aftertouch events

Program Change

Suppresses all program changes

Channel Pressure

Suppresses all channel pressure events

Pitch Bend

Suppresses all pitch bend events

MIDI Clock

Suppresses all MIDI clock events

### Sys-ex Decoder/Encoder/Decoder

#### Sysex Decoder

Match Sysex:



Send on channel:

Controller Kind:

Controller Number:  Value:  ☒ Suppress Sys-ex

#### Sysex Encoder

Channels:

Controller Kind:

Controller Number:  Value:  ☒ Suppress

Generate Sysex:

#### Sysex Patcher

Decode Sysex:



Encode Sysex:

These filters are described in detail in [the Sys-ex Patterns guide](#)

### Transpose



**Transpose**

Apply to Channel:

None Omni 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Key Range: 36 - C1 to 60 - C3

Transpose: +1.0 = 1 octave

New Channel: Same

The Transpose filter item transposes a range of notes up or down by a specified interval, and optionally changes the channel of those notes.

#### Apply to Channels

The MIDI channels this filter should be applied to.

#### Key Range

The range of notes to be transposed.

#### Transpose

The interval to transpose notes by.

#### New Channel

Optional setting to also change the MIDI channel number of transposed notes.

Notes outside the specified key range are not transposed, nor is their channel changed.

The ability to change channel is useful for creating keyboard splits, although this is now better covered by the new Keyboard Split filter item.

Also consider the related filter types [Keyboard Split](#) and [Key Range](#).

## Velocity Gate

**Velocity Gate**

Apply to Channel:

None Omni 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Velocity Range: 64 To: 90

Velocity Delta: -30 New Channel: Same

☐ Notes only ☒ Suppress out of range notes ☐ Passthrough original

The velocity gate filter can be used to remap notes according to their velocity. This can be used for example to play one instrument with soft notes and another with loud notes.

The keyboard range filter can be used to either remap notes to a new MIDI channel, or to simply suppress notes outside the specified velocity range.

#### Apply to Channels

The MIDI channels this filter should be applied to.

#### Velocity Range

Specifies the velocity range to remap

#### Velocity Delta

Specifies how much to adjust the velocity of remapped notes by

#### New Channel

Specifies a new channel number for all notes matching the specified keyboard range

#### Notes Only

When selected, only remaps note events. When cleared, all other events are also remapped.

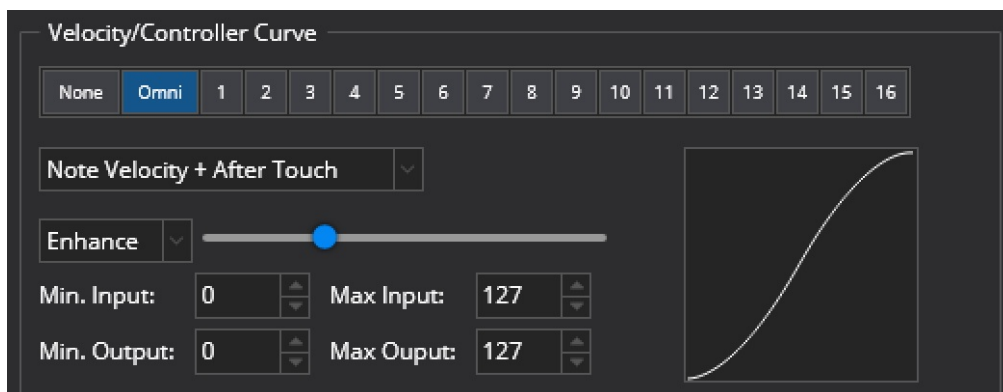
#### Suppress out of range notes

When selected all notes outside the specified velocity range are suppressed. When cleared, notes outside the specified velocity range are passed through the filter unaltered.

#### Passthrough Original

When cleared, notes that are remapped are modified and subsequent filter items only see the modified event. When selected, notes that are remapped generate a new event and the original event is passed to the next filter item unaltered

## Velocity/Controller Curve



The Velocity Curve filter item manipulates the values of velocity, controllers, aftertouch, channel pressure and pitch bend events.

### Apply to Channels

The MIDI channels this filter should be applied to

### Event Kind

The kind of MIDI event the curve should be applied to

### Curve/Enhance

Controls the shape of the translation curve

### Curve Amount

Controls the extent of the translation curve

### Minimum Input Value

The input value that should be mapped to the minimum output value

### Maximum Input Value

The input value that should be mapped to the maximum output value

### Minimum Output Value

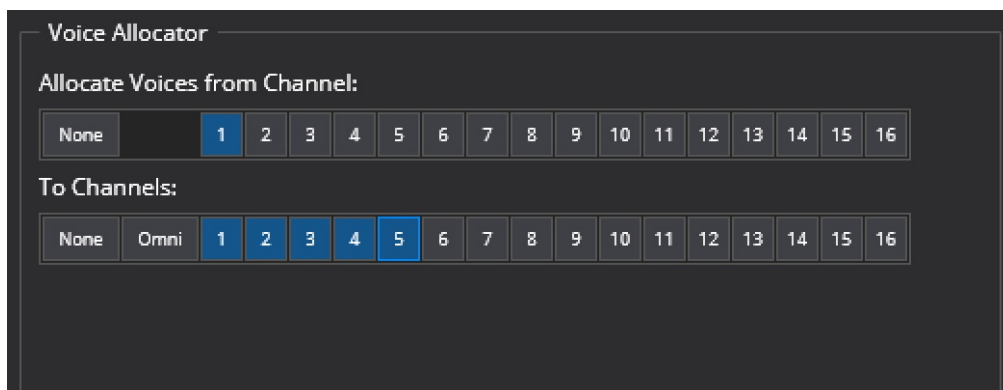
The minimum output value

### Maximum Output Value

The maximum output value

The graph can be used to visualize the type of translation being applied to note velocities. The X-Axis represents the input values, the Y-Axis represents the output values.

## Voice Allocator



The Voice Allocator maps notes out on different channels, and is intended for playing polyphonic on a monophonic synth by having several instances of the synth made available on different channels.

### Allocate Voices from Channel

The MIDI channel this filter should be applied to

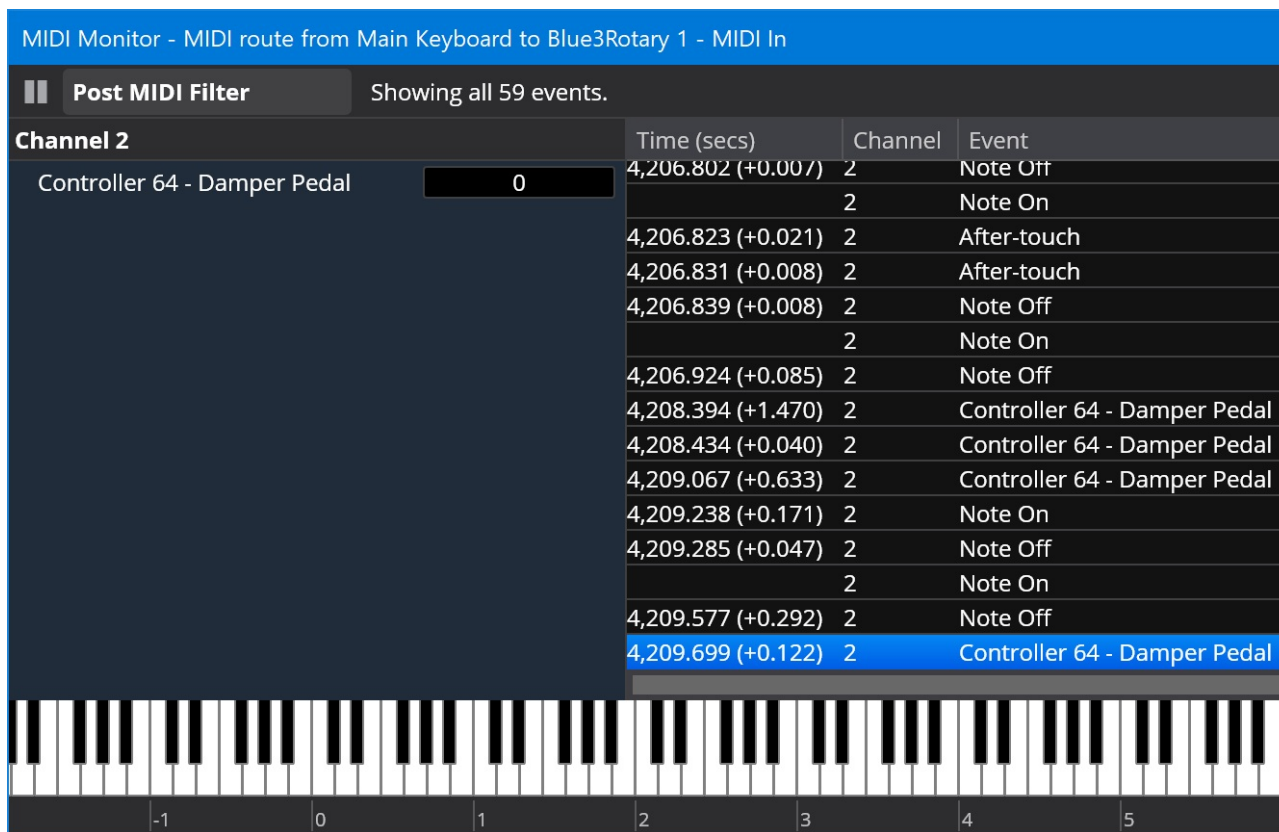
### To Channels

The set of channels that notes should be allocated to.

Notes are allocated on the available channels picking the least recently used inactive channel. If all channels are active, then the least recently channel is used after sending a note-off event to release it.

## MIDI Monitoring

Sometimes it's useful to be able to check the MIDI events passing through the system. To support this, Cantabile includes a simple MIDI monitor - a popup window that displays MIDI events in real-time.



The MIDI Monitor can be activated at many different places in the MIDI processing pipeline. To show a MIDI monitor right click on a port, route or object and choose MIDI Monitor.

Objects with MIDI filters will let you choose whether to monitor the MIDI events before or after the filter.

## Settings

To control which events area shown in the MIDI monitor, click the menu button at the top right of the MIDI Monitor window and choose "Settings".

MIDI Monitor Settings

Channels:

None

Omni

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

Show MIDI Events

☒ Controllers:  
0-127

☐ Not Controllers:  
0,32,6,38,98-101

☒ Note On

☒ Note Off

☒ After Touch

☒ Program Change

☒ Channel Pressure

☒ Pitch Bend

☒ Sys-ex

☒ MIDI Clock

Other

☒ Enable Filtering

☒ Show Keyboard

☒ Show Channel States

☐ Save these settings as the default

Show Decoded Events

☐ Fine Controllers:  
0-31

☒ Not Fine Controllers:  
0,6

☒ Banked Program Changes

☒ RPN Parameters (Fine)

☐ RPN Parameters (Coarse)

☒ NRPN Parameters (Fine)

☐ NRPN Parameters (Coarse)

☒ Master Volume/Balance Sys-ex

☒ MMC Events

OK

Cancel

## Global Transpose

*Cantabile Performer Only.*

Cantabile supports a global transpose settings. To access the transpose settings either:

- From the Control menu, select Global Transpose, or
- Click the Transpose button on the Input Ports slot (which also shows the current transpose setting)

State	Name/Source	Preset/Destination	Control
⋮	Input Ports		
⋮	✓ Main Keyboard	Blue3Rotary 1 - MIDI In	Omni
⋮	Add Route		
⋮	Output Ports		

The transpose settings window appears as a small popup:

Transpose

Global Transpose:

+0.2 = 2 semitones

Learn

Close

This setting controls the song wide transpose setting. It is saved with the song file and affects all MIDI routes in the loaded song and referenced racks.

The transpose setting can be controlled by states and this behavior enabled by default. To disable state control, select the Input Ports slot and remove the checkmark next to "Transpose" in the state behavior panel. (see [here](#) for more).

## Excluding MIDI Routes from Transpose

You can prevent a MIDI Route from being affected by the global transpose setting by opening the route's options and selecting "Ignore Global Transpose Settings".

This can be useful in two situations:

1. For routes to percussive instruments that shouldn't be transposed.
2. For routes where the transpose setting has already been applied.

## Notes

- Notes are transposed after any input port MIDI route split or key range settings have been applied (this allows adjusting the transpose settings without affecting split points).
- The transposition setting is taken into account by Cantabile's held note tracking so changing it while notes are held won't result in stuck notes.
- The Input Ports slot shows the current transposition amount as .. eg: "+1.0" = up one octave while "-0.2" = down one tone.

## Audio Engine Options

The Audio Engine options page provides settings that control various important aspects of Cantabile audio processing:

Options

General

Audio Engine

Audio Ports

MIDI Ports

Plugin Options

Recording

Startup/Shutdown

File Locations

Keyboard & Controls

Colors

Hot Keys

External Tools

Miscellaneous

Diagnostics

About

Audio Driver

Audio Driver:

ASIO - ASIO4ALL v2

Control Panel

Sample Rate:

44,100Hz

▼

Buffer Size:

256

▼

☐ Shared Mode (allows other programs to use same sound card)
 ☐ Double Buffered Audio

Buffer Duration:

5.80ms

Audio Engine

Number of Audio Threads:

1 - disable multi-core support

▼

Multi-processor Mode:

Aggressive

▼

Thread Affinity Mode:

Forced

▼

Prevent Memory Paging:

Disabled

▼

Sample Rate Conversion Quality:

High

▼

Switch to Power Plan:

Ultimate Performance

▼

☐ Enable Output Limiter, threshold:
 

80

▲▼

%

☐ Double precision (64-bit) audio

OK

## Audio Driver

The audio driver drop down lets you choose which audio driver Cantabile will use for audio processing. Cantabile supports ASIO drivers for full audio input/output support and WASAPI for output audio only.

In general we recommend using the ASIO driver that was shipped with your sound device. If no such driver was provided (or if its unreliable) you can use the general purpose ASIO driver [ASIO4ALL](#) which works well with Cantabile.

Most audio drivers will take exclusive control of the associated sound device making that device unavailable to other programs. This is because exclusive access provides the best possible latency and audio performance. If you need to use another audio program while Cantabile is running you can either:

1. Stop Cantabile's audio engine (the power button at the top right of the main window) while using the other program
2. Check if your ASIO driver supports a shared mode. (some do, not all)
3. Choose a WASAPI driver and set the sample rate drop down to "Shared Mode".

The third option is a good choice if you want to play along with other music played from your computer (eg: playing along with a YouTube video).

Most ASIO drivers have additional settings that can be accessed via the "Control Panel" button.

## Sample Rate and Buffer Size

The sample rate and buffer size options control the size of each audio buffer processed by Cantabile. The available choices

here are dictated by the capabilities of the selected audio driver.

In general:

- The higher the sample rate the better the quality, but at the expense of additional processing load.
- The higher the buffer size, the less chance of audio drop outs but at the expense of possible audio drop outs.

In general for live performance work we recommend setting the sample rate 44,100Hz and buffer size to 256 samples. This provides a good balance between acceptable latency and sufficient headroom to not cause audio drop outs.

For a detailed explanation of how to configure these options see our free eBook [Glitch Free](#).

## Double Buffered Audio

When the double buffered audio option is enabled, the audio engine introduces an additional audio buffer between the audio driver and the audio engine.

This can work particularly well with some poorly designed audio drivers that consume a significant portion of the audio cycle before passing control to Cantabile. It also provides an extra cushion to deal with small unexpected stalls in the audio processing.

Note that the extra layer of buffering also causes additional latency - effectively doubling the buffer size.

Some considerations:

- If you find you're getting audio drop outs when processing load is still below 100% (eg: some drivers can cause dropouts even with load as low as 85%) then using double buffering might be a good option.
- You can experiment with this option by enabling it and then halving (or at least significantly reducing) the buffer size. If you get acceptable performance in this configuration then using this option might be a good choice as it provides that extra cushion for unexpected stalls.

## Number of Audio Threads

Cantabile's audio engine can process multiple simultaneous tasks on separate threads. Normally you should set this option to the number of physical processor cores supported by your computer. Don't set this to the number of virtual hyper-threading cores - this almost always has detrimental affects on audio processing.

Other considerations:

- Some plugins (particularly older, non-supported plugins) have issues with multi-core processing. Setting the number of audio cores to one effectively disables multi-core processing and might eliminate problems with these plugins.
- If you're using plugins that have their own multi-core processing support, you may get better performance by reducing the number of Cantabile audio threads - free up some for plugin processing.

Again, refer to [Glitch Free](#) for a more detailed discussion of this topic.

## Multi-Processor Mode

As mentioned above, some plugins have issues with multi-core processing. This option controls how aggressively Cantabile tries to schedule plugin processing to different cores. If you find issues when running multiple instances of the same plugin causing crashes or other unexpected behaviour you can try setting this option to "Compatible".

If compatible mode resolves the issue, please [report the plugin](#) that's causing the problem.

## Prevent Memory Paging

This option can help on old machines (pre-Windows 10) that don't have sufficient memory to reliably run large sample based plugins. In general this option should no longer be used.

## Sample Rate Conversion Quality

This option controls the sample rate conversion of audio file playback and is only used when playing audio files with a sample rate different to the currently selected audio engine sample rate.

## Switch to Power Plan

When running any real-time audio software you should switch the selected Windows Power Plan to the "High Performance" plan. To save manually doing this everytime you run Cantabile, you can select a power plan here and Cantabile will automatically activate it when the audio engine starts and set it back when the engine is stopped.

## Enable Output Limiter

To avoid hard clipping of output audio, you can use Cantabile's audio limiter to automatically reduce loud signals back into range.

## Double Precision (64-bit) Audio

Select this option to enable 64-bit floating math processing for audio signal processing. Generally for live performance work this is excessive and should be left turned off. See [Glitch Free](#) for a more detailed discussion of this topic.

# Bindings

*Cantabile Solo and Cantabile Performer Only*

Bindings let you control nearly all settings in Cantabile (and any loaded plugins) from an external MIDI device like a keyboard, control surface or drum pad. Cantabile supports many different MIDI controller events and many different controllable elements within Cantabile.

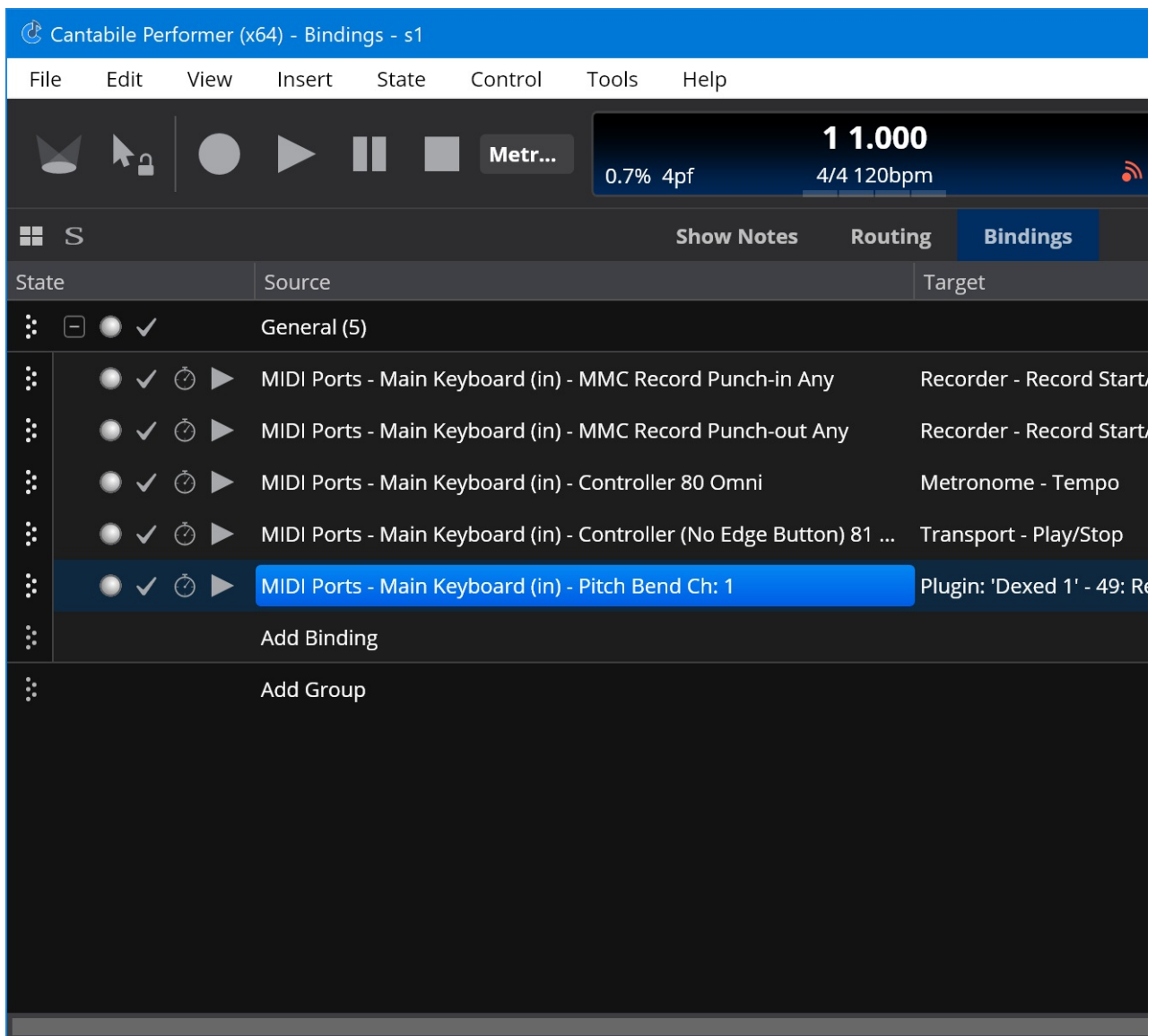
Cantabile Performer also supports "reverse bindings" where a setting inside Cantabile can be reflected to an external MIDI device so that whenever the setting changes an appropriate MIDI message is sent.

*Note: "Bindings" is the new name for what was called "MIDI Assignments" in earlier versions of Cantabile*

## Bindings Editor

Bindings are edited using the Bindings tab in Cantabile's main window:





You can switch to this tab either with the mouse, or by pressing Ctrl+D or by using Ctrl+Tab/Ctrl+Shift+Tab.

Double clicking on a binding brings up the binding editor panel for that binding:

Edit Binding - Bindings - General - #3

Source

Learn...

Object: MIDI Ports

Point: Main Keyboard (in)

Event: Controller

Controller: 80 Ch: Omni

Routing Mode: Continue

Target

Object: Metronome

Point: Tempo

Mapping

Mapper:

Prevent Jumps:

Source Range:

Target Range:

Out of Range:

Curve Kind:

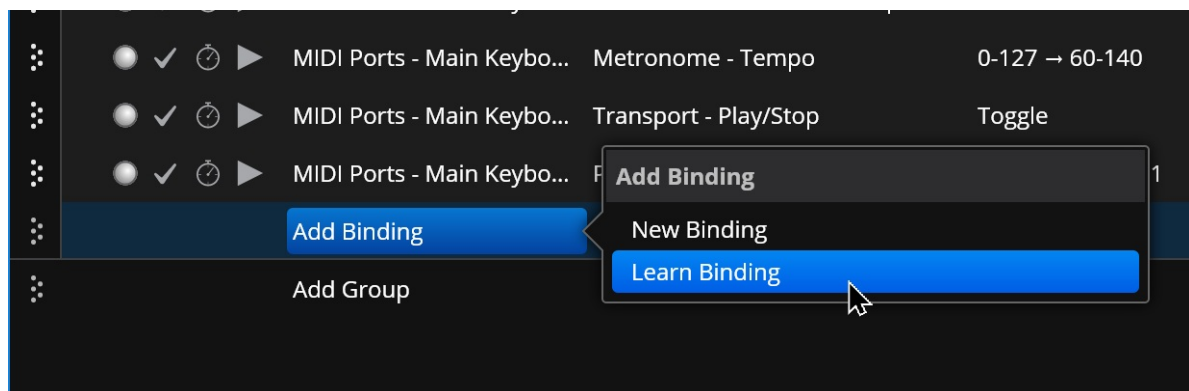
Curve:

Comments

☐ ☒ Enabled ☐ Bi-directional
Timing...
Test

## Learning Binding Source

The easiest way to create a new binding is to let Cantabile "learn" it. Click the "Add Binding" button and choose the "Learn Binding" option.

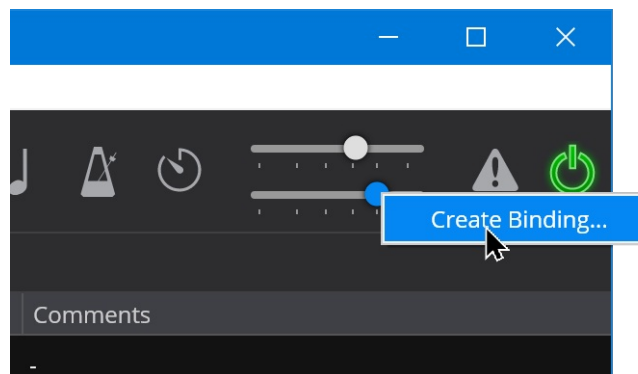


A window will appear that shows all incoming MIDI events. Move or press the controller that you want to bind and it should appear in the list. Double click the event you want to bind and a new binding with that source controller will be created.

You can then use the binding editor to configure the rest of the binding.

## Learning Binding Targets

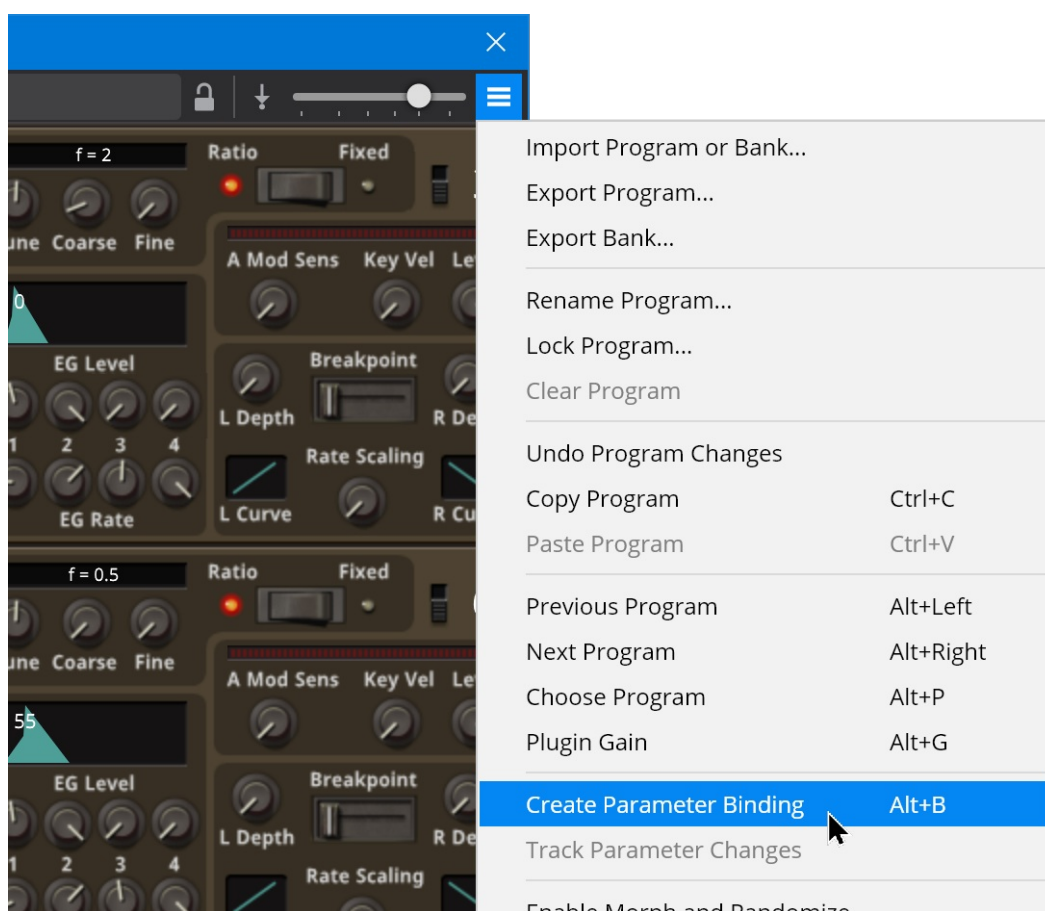
The other way to create a binding is by right clicking on the target setting you want to bind to. For example to create a binding to the master gain setting, right click on it and choose Create Binding:



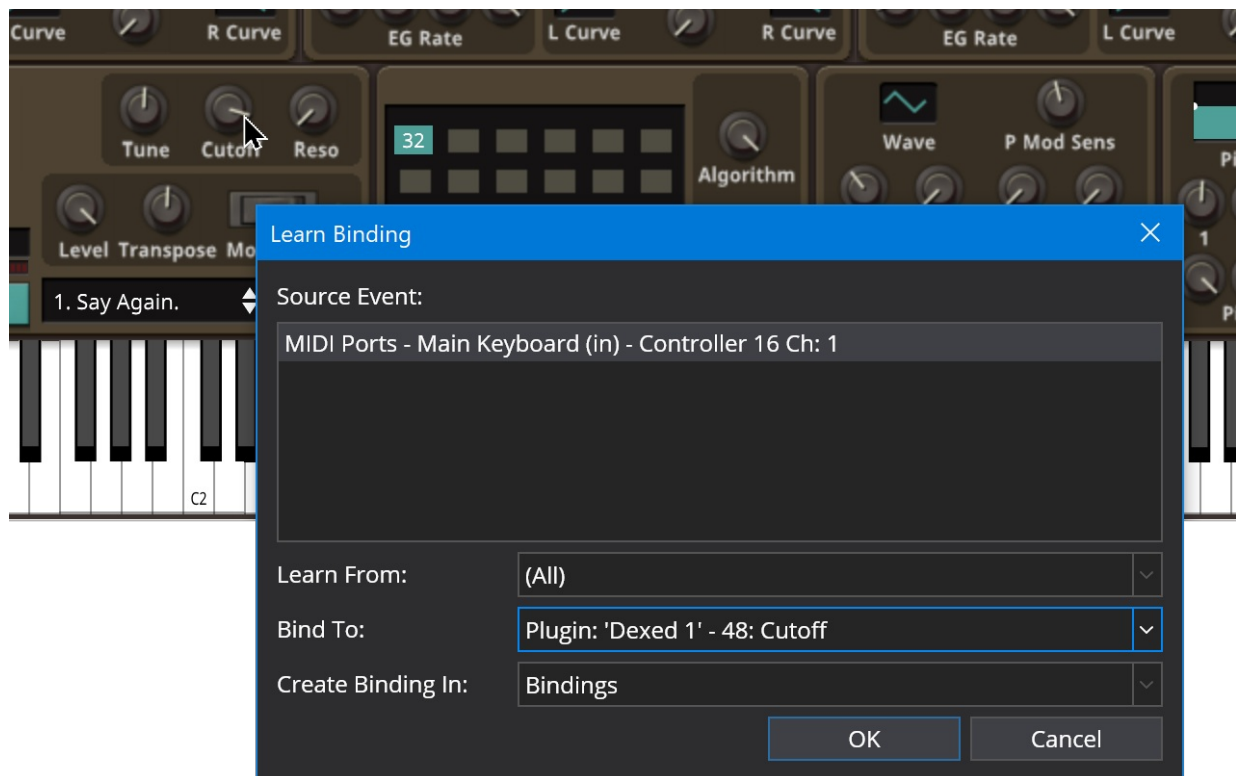
You'll then be prompted to move the control you want to bind to with the appropriate target binding point selected:

## Learning Plugin Parameter Binding Targets

To learn a binding to plugin parameter, open the plugin's editor and choose Create Parameter Binding:



While the Learn Binding window is open you can move any setting in the plugin editor and the associated parameter will be selected in the Bind To drop down. Here the "Cutoff" knob was turned resulting in the "Cutoff" parameter being selected:



## Binding Settings

Each binding has three main groups of settings:

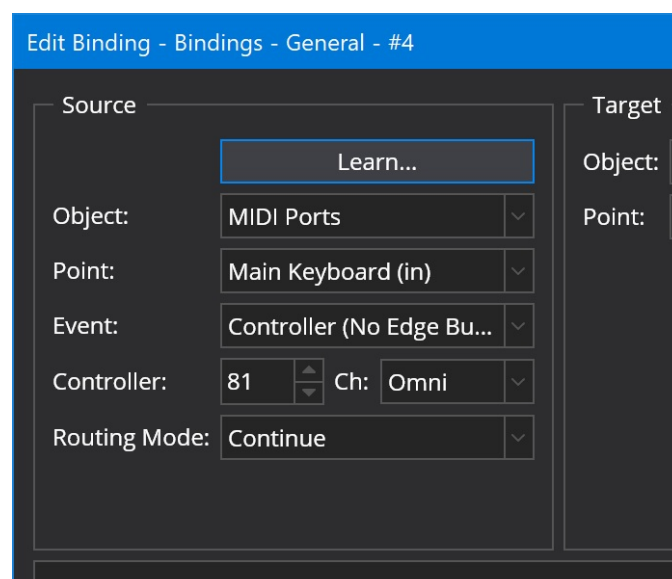
- The Source Binding Point - the setting, event or action that triggers the binding
- The Target Binding Point - the setting or action invoked when the binding is triggered
- The Mapper - how values from the source binding point are mapped to the target binding point

## Binding Points

The source and targets of a binding are specified as a "Binding Point" which is specified using an "object/point" concept:

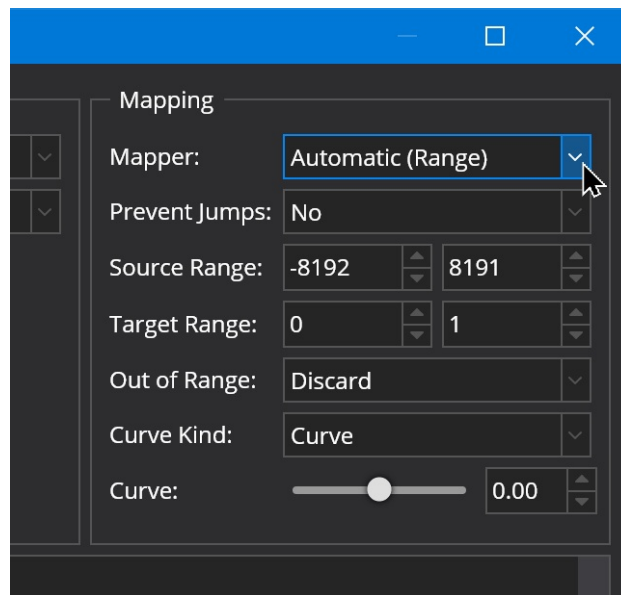
- The "object" is top level item that is triggering or receiving the event, eg: "Recorder"
- The "point" is the setting or event on the object that triggers or receives the event, eg: "Auto Record Mode"

Often a binding point will have additional parameters. For example, MIDI binding points have additional parameters that specify the MIDI event, controller, channel that trigger or will be sent by the binding point



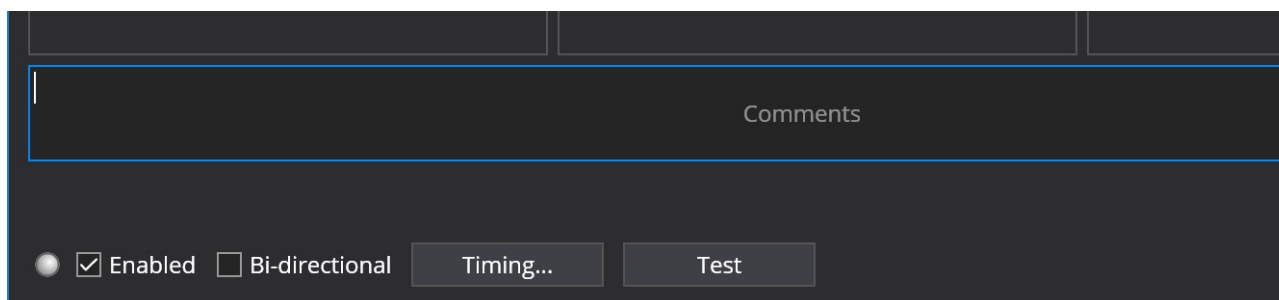
## Mapper

Mappers let you control how values are mapped between the source and target values. The set of available mappers changes depending on the selected source and target binding points.



## Other Settings

Besides the Source, Target and Mapper bindings also support the following settings:



### Comments

Type your own notes and comments here

### Activity Indicator

The LED indicator lights up when the binding is active - ie: when its responding to changes from the source.

### Enabled Checkbox

Enables or disables the binding

### Timing

Allows configuring delays and scheduling that control when the target side of the binding should be invoked. (Cantabile Performer v4 only). See below for more information on configuring these options.

### Test

Manually invokes the target side of the binding (for testing)

## Jump Prevention and Rotary Encoder (Relative CC) Support

See [Binding Jump Prevention and Relative \(Rotary\) Encoder Support](#)

## MIDI Routing Modes

MIDI Source binding points have a routing mode that determines how processing proceeds after the binding has been matched to an incoming event:

### Continue

The original event continues to be routed as if the binding didn't exist.

### Suppress

The original event is suppressed. Subsequent bindings and other MIDI targets won't receive the event. This mode is useful when binding specific notes to action (eg: binding the lowest key on your keyboard to move to the next song)

## Delays and Scheduling (Cantabile Performer)

In Cantabile Performer you can configure a delay between when a binding is triggered and when the target side of the binding is actually invoked.

Binding Delays and Schedule

Schedule

Before executing trigger, delay for:  
0 (milliseconds)

After executing trigger, delay for:  
0 (milliseconds)

Schedule this binding's delays with:  
Nothing (simple pre-delay only)

☒ Pause song and state load operations until complete

Re-trigger

Re-trigger Limit (milliseconds)  
0

The re-trigger limit controls how much time must elapse before this binding can be re-triggered

OK  
Cancel

The "before delay" is a delay between the time of the triggering event and the time the target side of the binding is invoked.

The "after delay" is a second delay that will be introduced between the time this binding is invoked and the next delayable binding is invoked. If there are no other delayable bindings after this one then this setting has no effect. (This option is not available if the schedule setting is set to "Nothing")

Bindings can be scheduled in several different groups within which the delay timings are calculated relative to each other.

ie: the pre-delay causes a delay between the last currently pending binding in the group and the post-delay causes a delay between this binding and any other subsequent bindings in the same group. For example, suppose you had two bindings each with a 500ms pre and post delay. If these were configured in the same group and they were triggered at the same time, they would be invoked as follows:

- in 500 milliseconds the first binding would trigger (500ms pre-delay on the first binding)
- in 1500 milliseconds the second binding would trigger (500ms pre-delay on the first binding + 500ms post-delay on the first binding + 500ms pre-delay on the second binding)

A third binding the the same group with a pre-delay of 0 would trigger at 2000 milliseconds (because of the 500ms post-delay on the second binding).

A bindings delay group is set using the "Schedule this binding's delays with" drop down.

- Nothing - only the pre-delay is used and is scheduled from the time the binding is invoked. This is a simple delay unaffected by other bindings.
- Globally scheduled bindings - all bindings in all racks and songs with this group selected will be scheduled relative to each other.
- Other bindings in this song or rack - all bindings in the same rack or song with this group selected will be scheduled relative to each other.
- Other bindings invoked by the same event - all bindings invokes by the same triggering event will be scheduled relative to each other.

The "Pause song and state load operations until complete" option specifies what happens if certain major operations occur (such as song or state switch, engine shutdown etc...) while the binding is in the pre-delay period:

- When enabled, the operation will block until the binding has fired.
- When disabled, the binding will be canceled and wont fire.

For example suppose you have a binding configured to fire 10 seconds after a song loads. Then suppose you load the song and then immediately switch to another song (before the 10 second time out). If the binding is configured as blocking then

Cantabile will block the song switch until the 10 second time expires, the binding will then be invoked and then the new song will be invoked. If the binding is configured as non-blocking the delayed action will be canceled and the song switch will happen immediately.

## Sys-Ex Bindings

*Cantabile Performer Only.*

A binding can generate sys-ex data if the target is a MIDI port and the event kind is set to "SysEx". For details on setting up sys-ex bindings, see the [Sys-ex Expression reference](#).

## Global Bindings

Often you will want to configure bindings that work across all songs.

eg: it doesn't make much sense to create a binding in a song to load the next song in a set list since you'd need to create an identical binding in all songs for it work correctly.

For this reason, Cantabile has a "Background Rack" - a special rack that is always running in the background and whose primary purpose is for global bindings.

To create global bindings:

1. From the View menu, choose "Background Rack"
2. Switch to its Binding tab
3. Create bindings as per any other rack

Note that not all binding targets are available from the background rack. eg: you can bind to a specific plugin in a song because that song might not be loaded when the binding is invoked.

## Song, Rack and Background Rack Bindings

Each song and rack has it's own set of bindings and bindings are processed in the following order:

1. Song bindings
2. Bindings in racks loaded into the active song, in the order the racks are in the song
3. All other rack's bindings (except the background rack) but in an undefined order
4. The Background rack's bindings

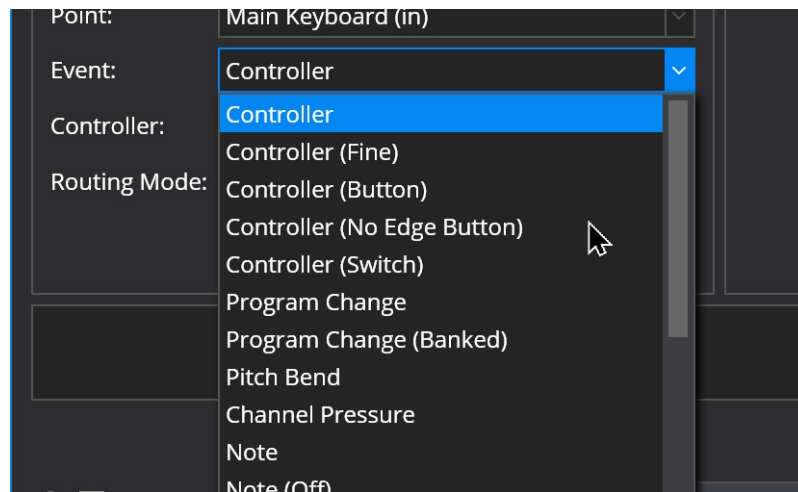
The most important part of all this is that the Song bindings take precedence over the Background rack bindings.

Let's say you have a button on an external controller that for all your songs you want it to perform some function. You could create a binding for that action in the background rack.

Now let's say for that for one particular song you want to change that button to perform some other action. If you create that binding in the song and set its routing mode to suppress—the song's binding will take precedence over the background rack bindings and effectively override it.

## Special MIDI Event Kinds

Most event kinds are self explanatory as they reflect the associated MIDI event. For MIDI Continuous Controller events (CC's) however there are a couple of different ways they can be used - depending mostly on the type of physical control they're originating from:



#### Controller

This is a normal MIDI CC event that can be assigned to a variable value (eg: a gain level)

#### Controller (Fine)

These bindings combine MIDI CC 0-31 with 32-63 to create a more precise 14 bit controller.

#### Controller (Button)

A momentary MIDI button that sends a value > 64 when pressed and < 64 when released. This kind of binding is edge-triggered in that it invokes the target when the value crosses from less than 64 to greater than 64.

#### Controller (No Edge Button)

Use this for MIDI buttons that send a non-zero value when pressed and no event or a CC value of 0 when released. ie: it triggers on any non-zero value.

#### Controller (Switch)

This is for non-momentary on/off MIDI buttons and switches that send a value  $\geq 64$  when on and  $< 64$  when off.

## Bindings to Routes

Unlike most other objects, routes do not automatically appear as assignable objects in the Bindings editor. This is because by default routes don't have a name and therefore can't be referenced.

To create an binding to an audio or MIDI route:

1. Select the route in Cantabile's main window
2. From the Edit menu choose "Rename", or press F2
3. Type a unique name for the route and press OK
4. Though the name won't show in the main window, the new name will appear in the Bindings
5. From the bindings editor you'll now be able to create bindings for the route

All routes support binding to the enabled/disabled setting. Audio routes also support bindings to the route's audio gain level.

## Bi-Directional Bindings

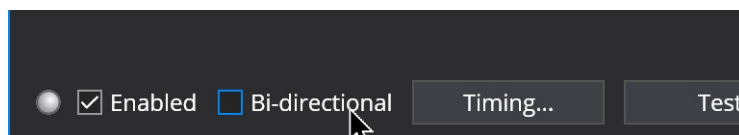
Cantabile Performer supports bi-directional bindings that establish a two way binding between the binding's source and target. Bi-directional bindings are generally used when binding from a motorized MIDI controller, or from a MIDI controller (knob) with an LED ring to provide feedback to the controller on the current state of the controller.

Bi-directional bindings have certain limitations, including:

- The source must be a MIDI source (ie: a MIDI input port)
- The source and target objects must have the same (or similar, see below) name
- The target must not be a MIDI target
- Not all MIDI event types are supported. Specifically, events that trigger an action (eg: Controller - Button) are not supported.
- The source side of the binding must have a specific channel selected (ie: Omni channel bindings aren't supported)

If the binding is configured as bi-directional, but the reverse binding can't be created due to one of the above reasons, an error message explaining the reason will be displayed when attempting to enable the bi-directional mode. The reason will also be displayed in the tooltip of the bi-directional binding button.

To enable bi-directional mode on a binding enable the "Bi-directional" checkbox in the binding editor:



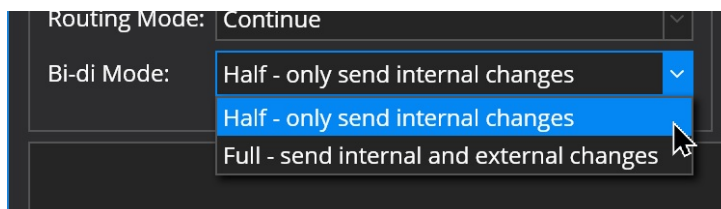


Bi-directional bindings are typically used as follows:

1. Go to Options → MIDI Ports
2. Make sure there is both a MIDI Input and MIDI Output port for the target device with the same (or similar, see below) names
3. Create a binding from the MIDI Input device's CC number to the target value to be controlled
4. Set the bi-directional mode - half for motorized faders, or full for LED rings

## Bi-Directional Binding Modes

Bi-directional MIDI bindings can be configured as either "half" or "full" bi-directional using the setting in the source column:



- *Half Mode* - the reverse binding is suppressed when the forward binding is changing the target value. This can be used with motorized faders if the fader jitters or doesn't move smoothly due to feedback from the reverse binding.
- *Full Mode* - the reverse binding is always active and is typically used with LED rings.

## Bi-Directional Bindings and Similar Object Names

As mentioned above, the source and target objects of a binding must be "similar" for bi-directional bindings to work. To find a similarly named object Cantabile uses the source object name with the following whole word, case sensitive replacements:

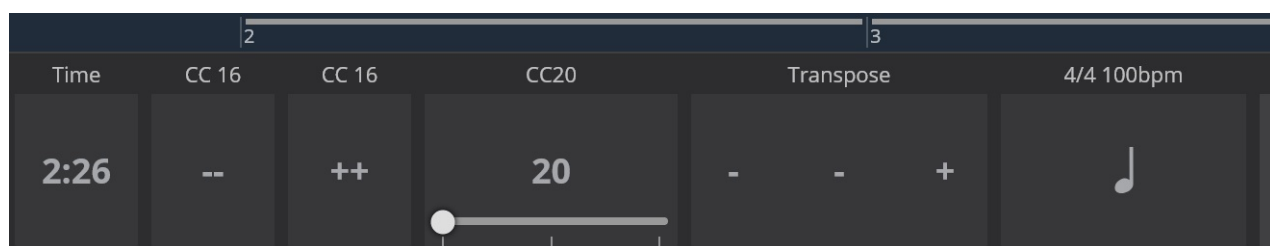
- "in" → "out"
- "In" → "Out"
- "IN" → "OUT"
- "input" → "output"
- "Input" → "Output"
- "INPUT" → "OUTPUT"

This allows for bi-directional bindings between port names like "Rack MIDI In" and "Rack MIDI Out"

## Controller Bar

*Cantabile Performer Only*

The Controller Bar provides large buttons intended for use during live performance to control key aspects of Cantabile as well as an array of user configurable buttons.



To show the Controller Bar, select "View" menu → "Controller Bar".

## Use Configurable Buttons

The left half of the Controller Bar contains a set of user-configurable buttons. By default they're configured to send MIDI program change events (from the on-screen keyboard device) but they can be reconfigured to a wide range of other functions.

To setup a button either right click on an existing button and choose "Customize", or choose "Add Button" to create a new button.

The 'Customize Button' dialog box is shown with the following settings:

- Tip: CC20
- Caption: 20
- Action: MIDI Controller
- Interaction Style: Popup Slider
- MIDI Channel: 1
- MIDI Controller: 20
- MIDI Minimum Value: 0
- MIDI Maximum Value: 127
- ☐ Auto-Home Value: 0

Buttons: OK, Cancel

The following settings are available:

- Tip - the text to display above the button
- Caption - the text to display on the button
- Action - the action to invoke when the button is pressed. Includes options to send various MIDI events or to invoke a built in Cantabile command.
- Interaction Style - See below.
- MIDI Channel - for MIDI actions the channel number of the event.
- MIDI Controller/Program/Parameter - depending on the selected event type the controller or parameter number to be sent with the event.
- MIDI Pressed and Released values - MIDI values for buttons when they're pressed and released, or for sliders changes to Minimum and Maximum value.

The interaction style determines how the button responds when it's pressed:

- Button - a simple command button - invokes the selected action when pressed.
- Momentary Button - a button that sends one value when pressed and another when released.
- Toggle Button - a button that sends a different value each time it's pressed.
- Popup Slider - a button that when pressed displays a popup slider.

## Dynamic Button Labels

The Tip and Caption fields of custom buttons can use [variables](#) to generate dynamic text. When combined with MIDI controller variables this can be used to reflect CC values on the controller bar.

## Built In Buttons

The right half of the Controller Bar displays 5 standard controls for common operations during live performance.

Each of these buttons can be optionally hidden if not needed by right clicking on the Controller Bar and clearing the check mark next to the selected item.

- Transpose Button - displays the current song's transpose setting and includes increment and decrement buttons to change the transpose setting. Clicking the middle of this button displays a popup to select a transpose.
- Tempo Button - displays the current song's tempo and flashes in time when the master transport is playing. Tapping this button sets the tempo according to how fast it's tapped.
- Song Picker - displays the currently loaded song in the set list. Arrow buttons can be used to select the next or previous song and tapping the song name displays a popup to select a different song.
- Song Part Picker - displays the currently loaded part in the current song. Arrow buttons can be used to select the next or previous part and tapping the part name displays a popup to select a different part.
- Continue Button - moves to the next song part. If currently on the last part of a song, moves to the first part of the next song.

## Resizing and Re-ordering Buttons.

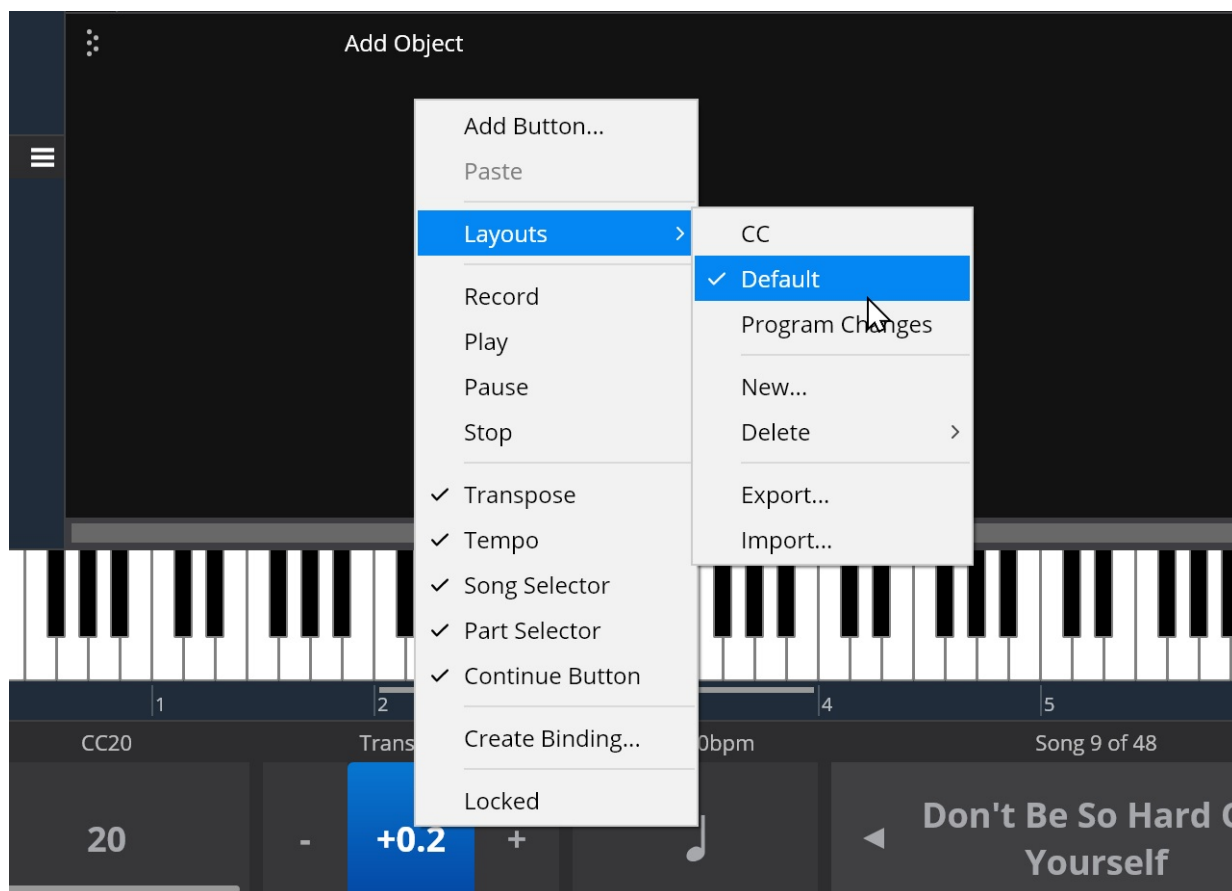
All the buttons in the Controller Bar can be resized by dragging the dividers between the buttons.

The user-configurable buttons can be re-ordered by right clicking a button a choosing "Move Left" or "Move Right".

## Layouts

The controller bar supports switching between various user defined layouts, where each layout stores the current set of custom buttons and the visibility of the built-in buttons. You can create as many layouts as you like.

To work with layouts, right click on the controller bar and use the Layouts sub-menu:



Tip: Live Mode stores which controller bar layout is active so you can have a different active layout for normal vs live mode.

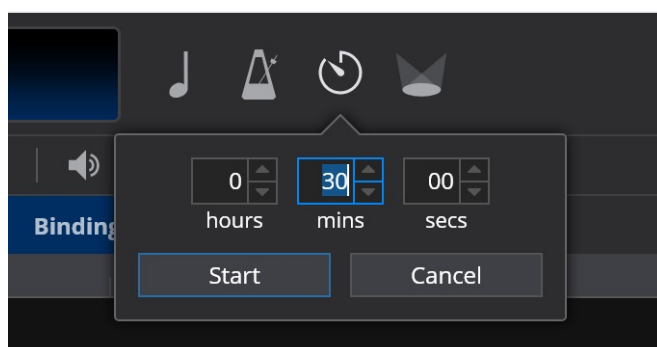
## Countdown Timer

*Cantabile Performer Only*

Cantabile's Countdown Timer is a general purpose timer that can be used for simple reminders like start of show, set breaks and practice sessions.

### Starting the Timer

To start the timer, click the timer icon, enter a time period and click start:



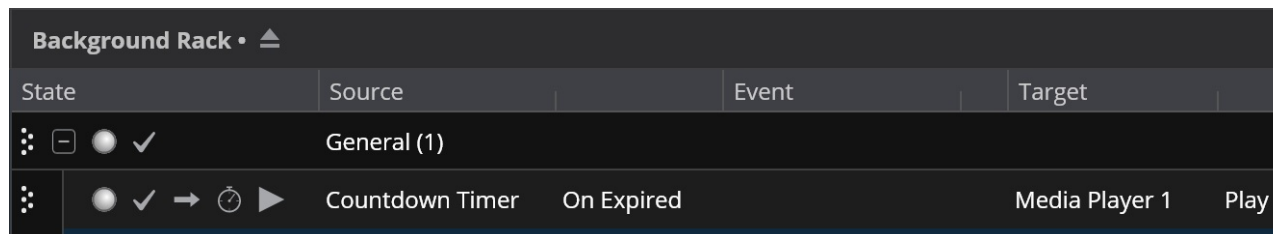
Note, if using a mouse you can enter a time by hovering over the fields and using the mouse wheel.

By default, the timer doesn't provide any notification when it finishes however you can use bindings to play sounds, display a message or perform other actions.

## Playing a Sound

To play a sound when the timer ends:

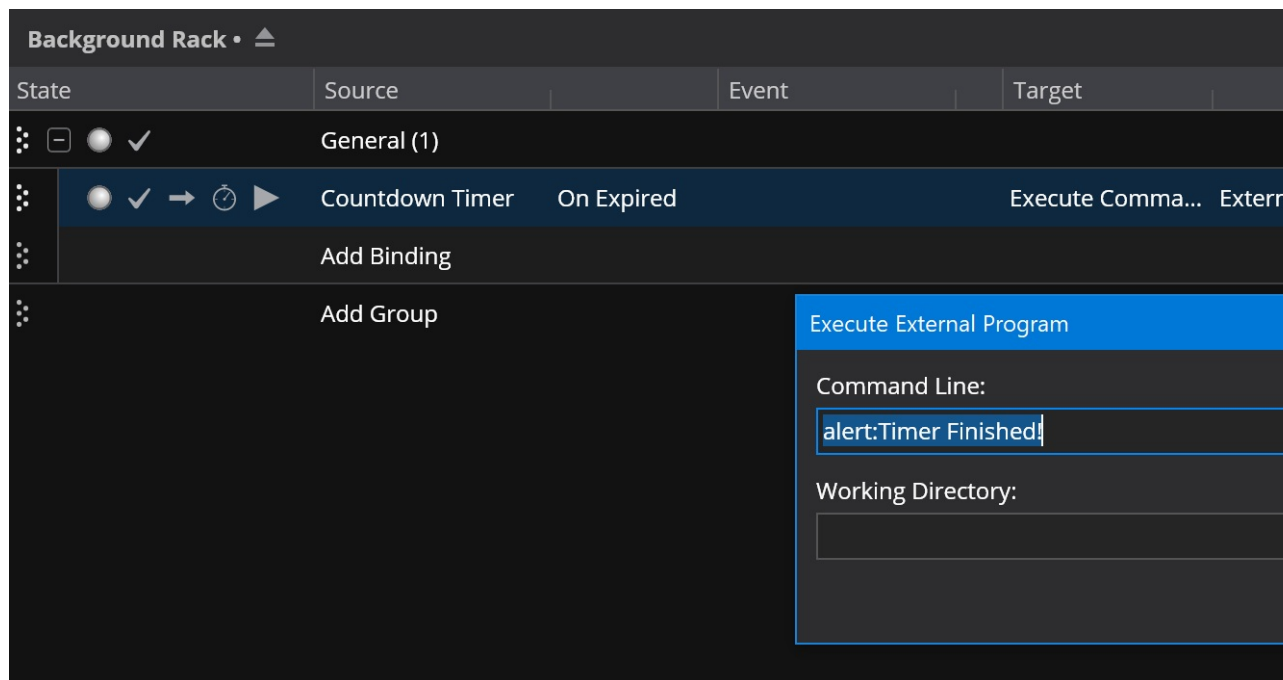
1. Open the Background Rack (View menu → Background Rack)
2. In the Routing tab, load a Media Player and load a file with the sound you want to play
3. Set the output of the Media Player to the output port the sound should be played (eg: you might like to send the sound to the Metronome port)
4. In the Bindings tab, create a binding from the Countdown Timer's "Expired" event to the the media player's Play command



## Displaying a Message Box

To display a message when the timer ends:

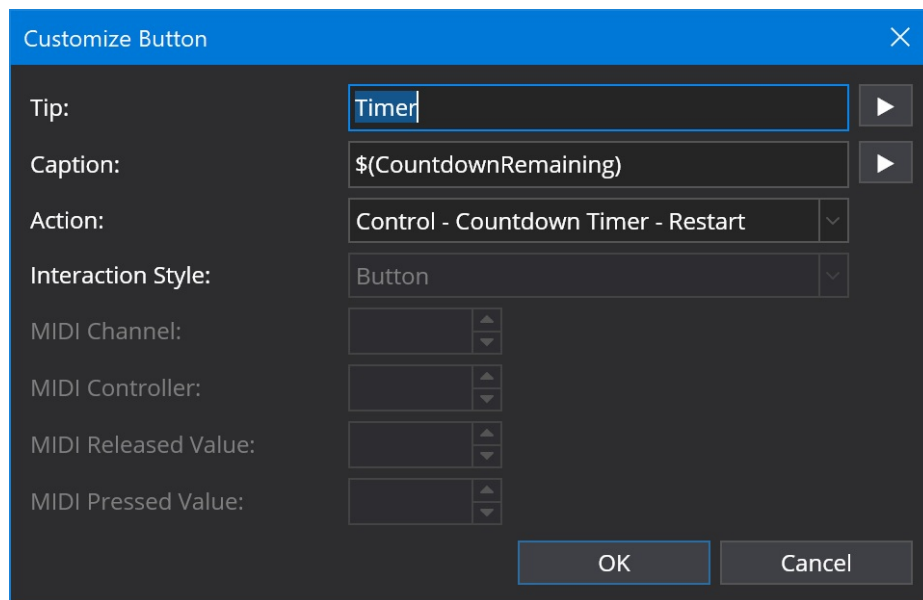
1. Open the Background Rack (View menu → Background Rack)
2. In the Bindings tab, create a binding from the Countdown Timer's "Expired" event to the Execute Command target
3. Set the Execute Command target to External Script, but use the prefix "alert:" to display a message box



## Displaying the Remaining Time

When the timer is running, the remaining time is displayed below the timer icon. You can also display the remaining time anywhere where you can enter a [variable string](#) with the variable `$(CountdownRemaining)`.

For example, to show the remaining time on a controller bar button, you could configure it like so:



## Other Bindings

The Countdown Timer supports other bindings points for remotely starting, pausing and stopping the timer and events for monitoring the current state of the timer.

See also: [Bindings](#)

## Interaction with Audio Engine

The countdown timer runs independently of the audio engine so it will continue to countdown even if the engine is started or stopped.

Note however that bindings only execute when the audio engine is running, so if the engine is stopped you won't get sounds or alerts as described above.

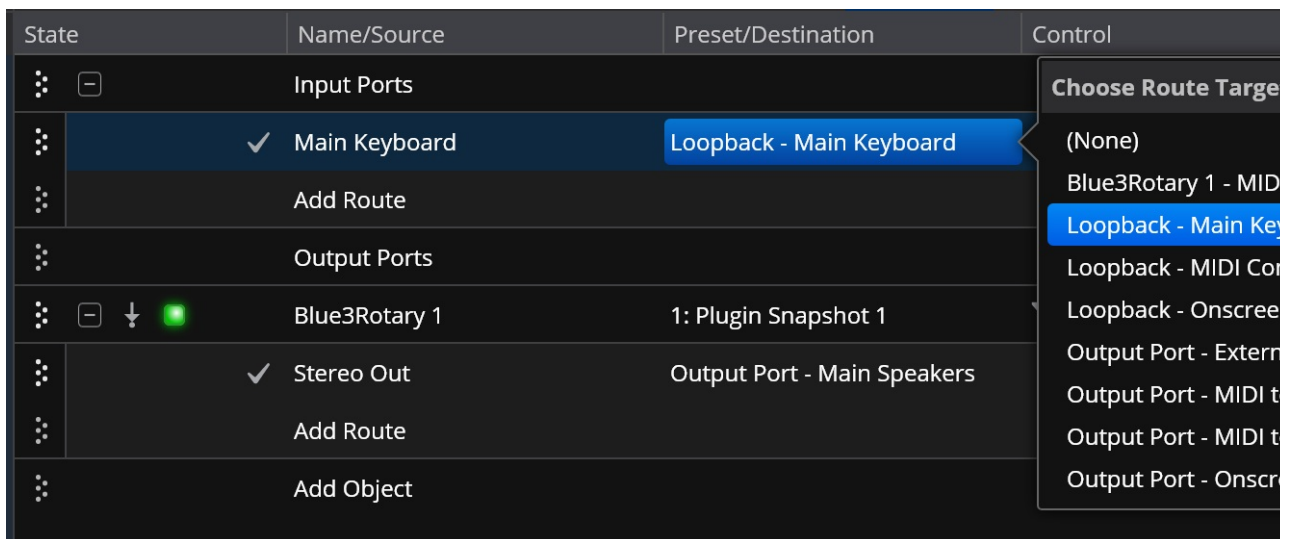
## Loopback Ports

Loopback ports provide the ability to send MIDI and Audio signals from within Cantabile back to an input port. ie: the signal is looped back from an output port to an input port.

By default loopback ports are disabled because they're no longer that useful. (They used to be a handy way to send MIDI signals to Cantabile's MIDI input processing where they could be picked up by MIDI bindings).

The feature still exists however and can be enabled in Options → Advanced → Show Loopback Ports.

When enabled, you'll notice that for every configured input port, an output Loopback port is available as a routing target.



In the above example, any MIDI sent from the media player would be routed back to the Main Keyboard input port and appear as if the events were coming from the Main Keyboard.

Care should be taken with loopback ports not to create feedback loops. Creating a circular loop of MIDI events can crash the program and create typical feedback sounds with audio loops.

## Media Players

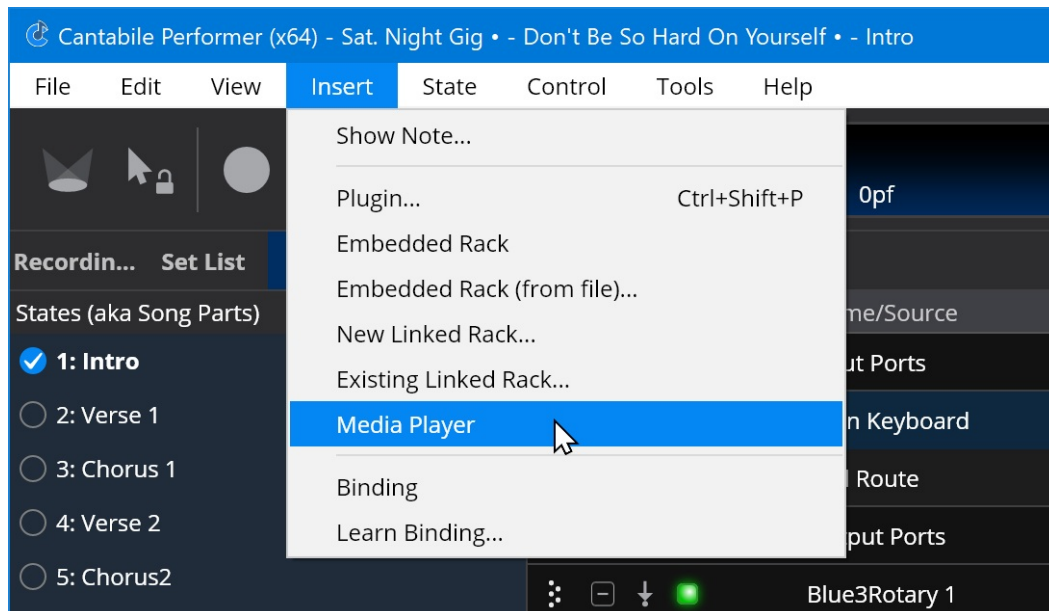
*Cantabile Solo and Cantabile Performer Only.*

Cantabile support playback of Audio and MIDI files. The following file formats are supported: .mp3, .wav, .flac and .mid or .midi.

This [video walkthrough](#) also demonstrates working with Media Players.

## Using a Media Players

To create a Media Player, from the Insert menu choose Media Player:

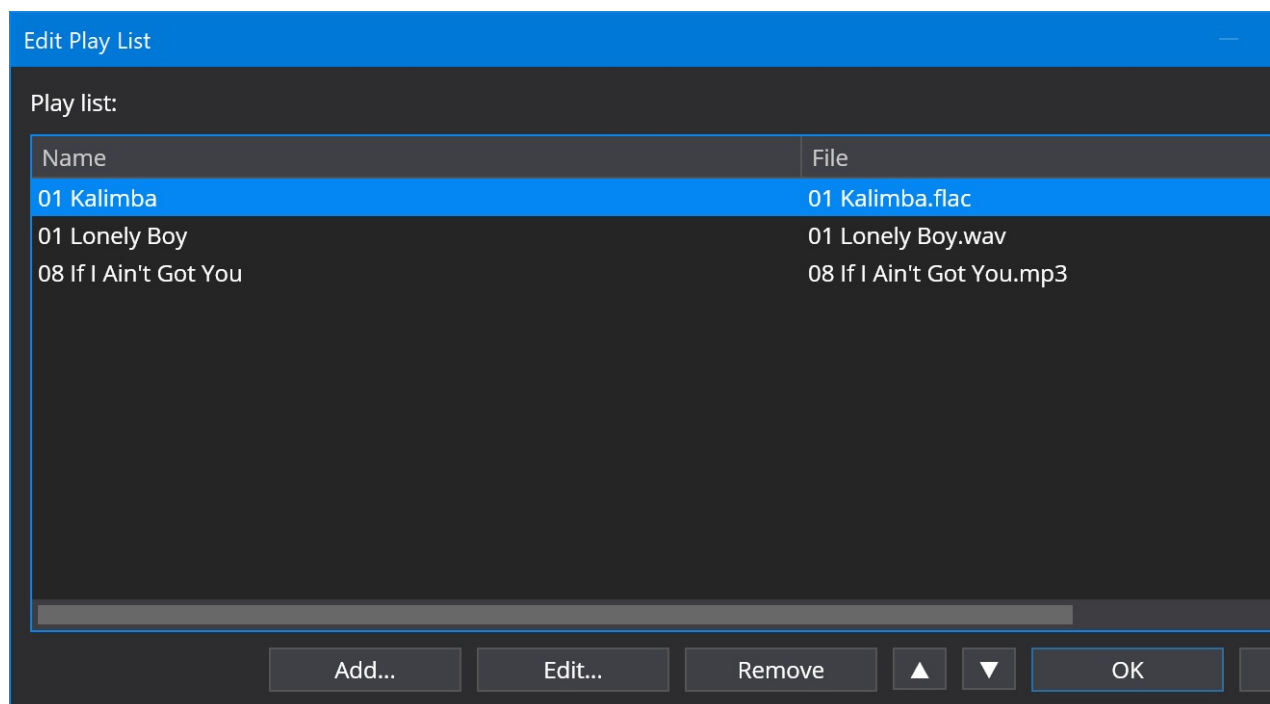


Once inserted, a new Media Player slot will appear in the main panel:



Each media player supports a playlist of media files. To setup the play list, double click on the media player and then click

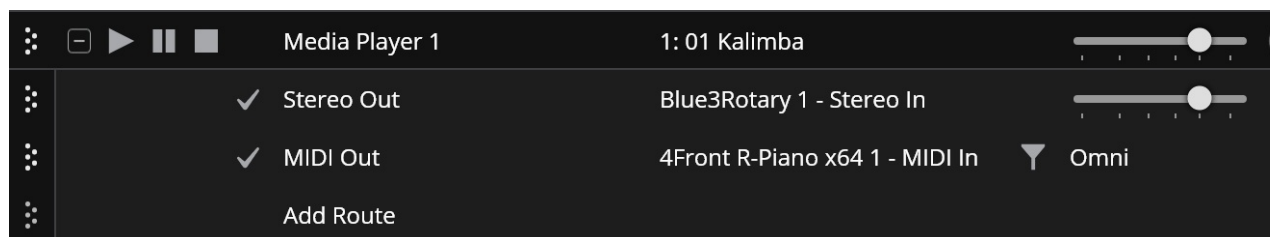
the Add button to load some media files:



The above example shows a playlist with two .mp3 files and three .midi files.

The next step is to configure routes from the media player to either output devices, or plugins. By default each media player is created with one stereo output port and one MIDI output port.

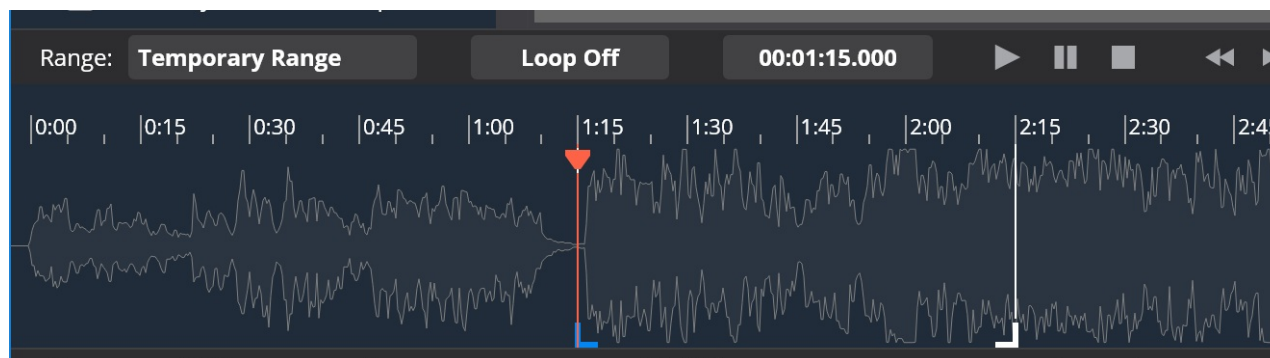
The following screen shot shows the MIDI output routed to a VSTi and the audio output routed to an effect plugin:



You can now use the player's transport buttons to play the selected file. The file selector drop down can be used to select which file is to be played.

## Timeline View

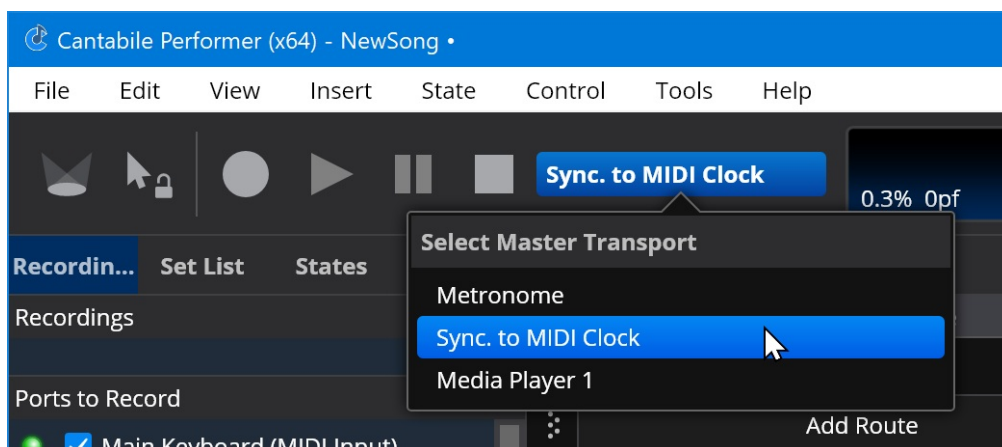
To view the contents of a media file, set play ranges and loop modes, use the [Timeline Panel](#).



## Master Transport Selection

The master transport provides the master timing information that other parts of Cantabile can synchronize to. For example the master transport provides timing information to any loaded plugins and is used to control MIDI output clock signals.

By default the master transport controls Cantabile's built-in metronome. When using media files however it is often desirable to use timing information from the playing file. To change the master transport, select the required transport from the drop down on the main toolbar:



The main transport buttons on the main toolbar always control whichever object is selected in the master transport drop down.

## Synchronization of Media Players with Master Transport

Media files can be synchronized to the master transport - which might be another media player, the metronome or the external MIDI clock.

When a media player is synchronized it starts and stops playing when the master transport starts and stops. Also synchronized media players track the current play position of the master.

To synchronize a media player, right click on it and choose one of the slave synchronization modes:

- Off - not synchronized
- Master - makes this media player the master transport
- Slave Realtime
- Slave Master

The difference between realtime and musical is the playback speed/tempo of the synced media file.

- Realtime - the synced media file plays at it's native speed and depends on the slave and master both having the same tempo in order to stay in sync. eg: A metronome set to a particular tempo and an audio file recorded at the same tempo.
- Musical - here the target MIDI player ignores the tempo information in the MIDI file and instead tracks the master transport. eg: when a midi file is synced to the metronome, changing the tempo of the metronome will cause the midi file to play faster/slower.

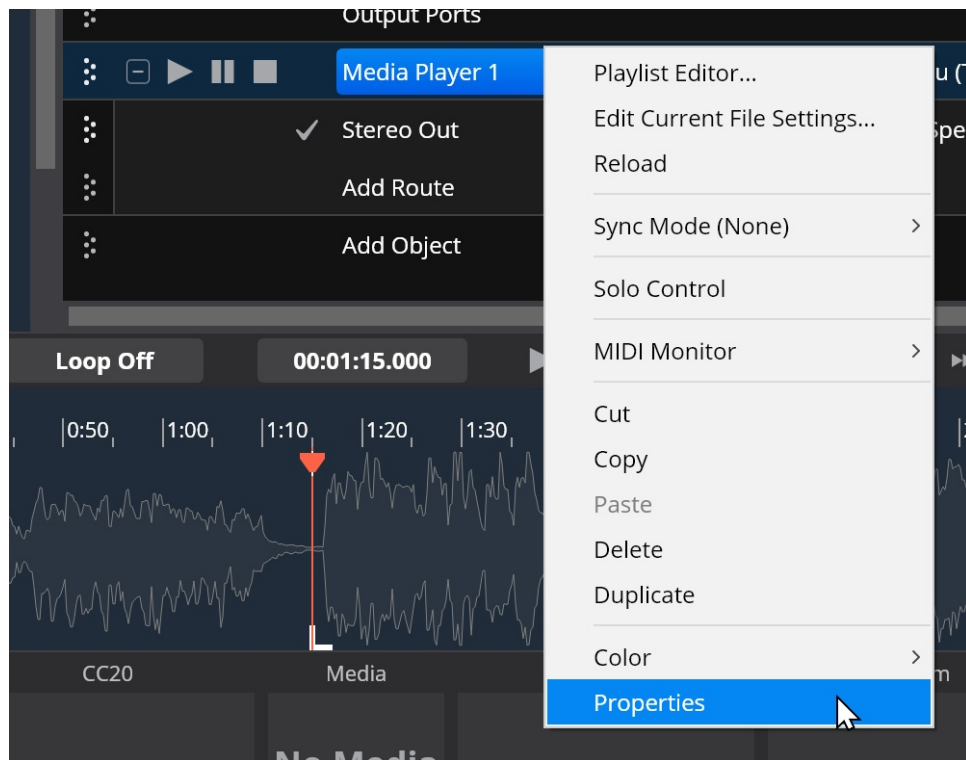
The Musical sync mode only works with MIDI files. If selected for an audio file, musical syncing will be used when setting the start position for playback, but during play the audio file always plays at its native speed.

## Customizing Audio and MIDI Ports

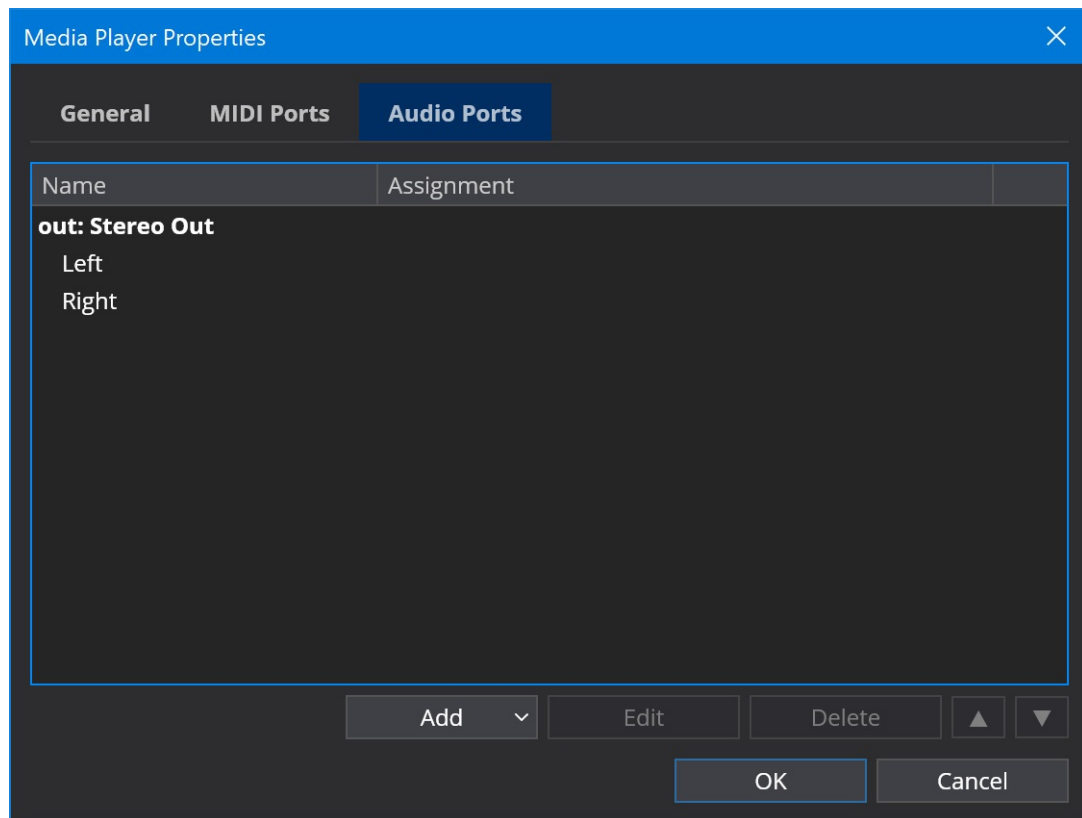
As mentioned above, each media player is created with a default set of MIDI and Audio output ports. You can however create new ports, or reconfigure the existing ones to suit your needs.

Right click on the media player and choose "Properties"

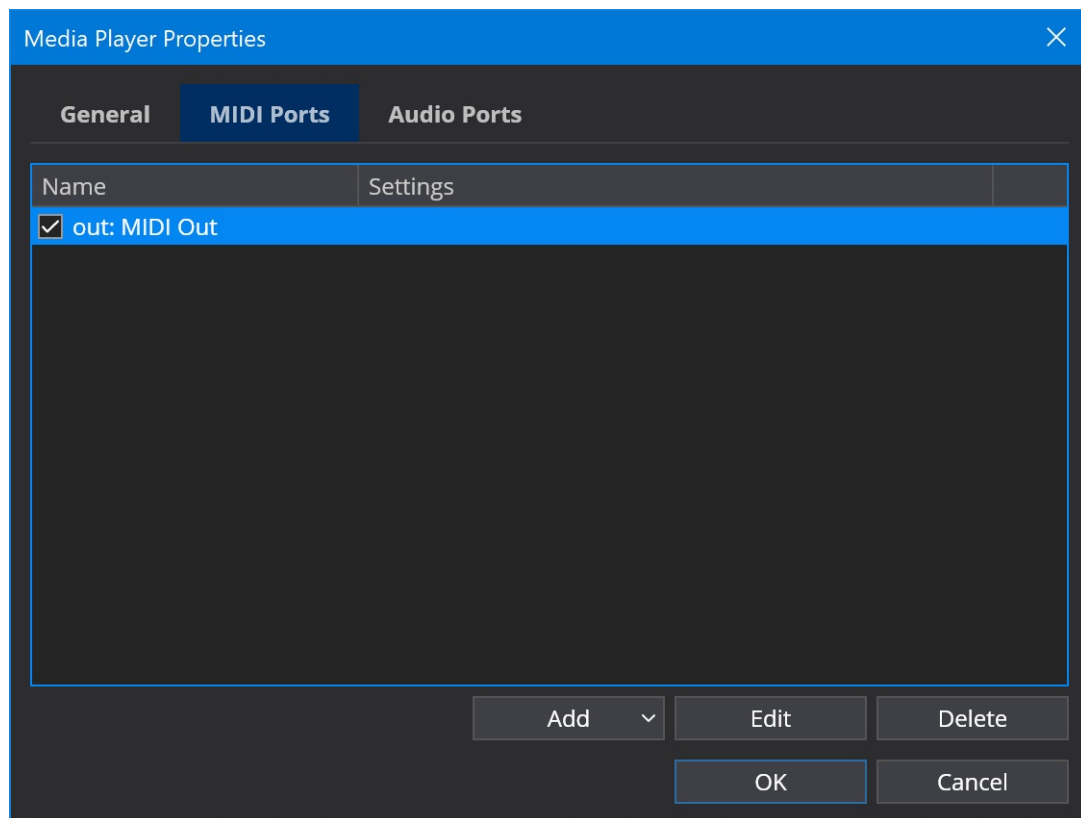




Configuring ports is similar to anywhere else in Cantabile where ports can be edited. Here's the audio port editor:



MIDI ports are similar:



After you've created the required ports you can route each port to different plugins or physical outputs.

## Editing Per-file Settings

Each media file managed by the media player has a set of properties that control how it's track and/or channels are mapped to the player's output ports as well as various tempo and timing settings

To edit these properties, double click the media player to open the Play List editor and then double click a file to edit it's assignments. You can also directly edit the currently loaded file's assignments by choosing "Edit Current File Settings" after right clicking on the media player slot.

Media File Properties

Media File

Name: 08 If I Ain't Got You

File: C:\Users\Brad\Music\Media\08 If I Ain't Got You.mp3

Browse...

Playback Settings

Tempo: 120

Time Sig: 4/4

Gain: 0.0 dB

Speed: 100.0%

Transpose: 0.0 = perfect unison

☒ Ignore Global Transpose Settings

Port Assignments

Source Channel	Port	Port Channel	
Ch: 1	Stereo Out	Left	
Ch: 2	Stereo Out	Right	

Add...

Edit...

Delete

▲

▼

OK

Cancel

Help

These settings perform the following functions:

#### Tempo and Time Signature

For audio files this defines the tempo and time signature of the audio and is used to provide musical timing information when the audio file is the master transport (see below). For MIDI files this information normally comes from the MIDI file itself however these settings can be used to override any embedded timing information.

#### Gain

Provides control over the output level of the media player for this file. For audio files the gain is applied directly to the output signal. For MIDI files the velocity of note events is adjusted to simulate volume gain.

#### Speed

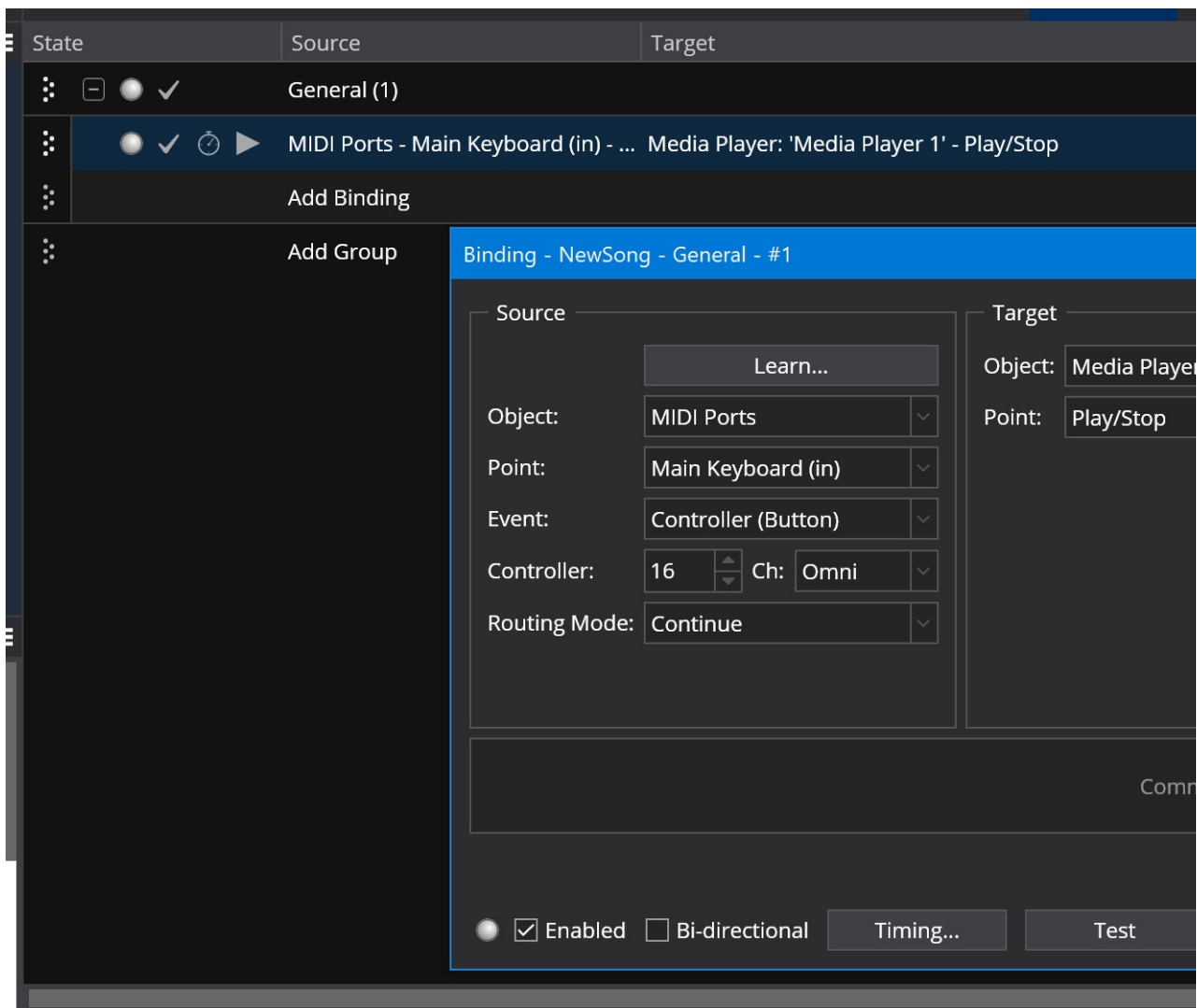
Controls the playback speed as a percentage of the original speed.

#### Port Assignments

Use this section to map audio file channels and MIDI file tracks to the media players output ports. You can create multiple assignments from each source track channel if necessary - each with different audio gain settings or MIDI track filters.

## Bindings

Media files can be remotely triggered from a MIDI control surface using [Bindings](#). For example the following binding starts/stops the media player each time a button with CC #16 is pressed on the "Main Keyboard" device.



## State Support (Cantabile Performer Only)

Media players can also be controlled via [States](#). States can be used to control the following settings between states:

- Selected File
- Gain Level
- Playback Speed
- Pan and Fade
- User Notes
- Slot Color

## Metronome

Cantabile includes a metronome that serves two main purposes:

1. As a general purpose metronome for practice
2. To provide timing information for plugins (eg: step sequencers, synced loops etc...)

## Showing the Metronome Toolbar

To show the metronome:

- Click the Metronome button on the main toolbar, or
- Press *Ctrl+M*

In either case, keyboard focus will be moved to the metronome.

To hide the metronome:

- Press the Metronome button again, or
- While the metronome has keyboard focus, press *Shift+Escape*.

You can also return focus from the metronome to the main plugin list by pressing *Escape*.

## Metronome Controls

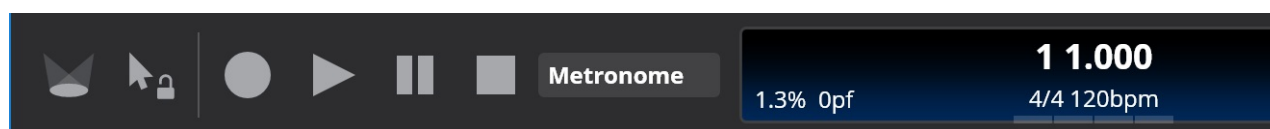


The metronome toolbar is divided into three sections: time signature, tempo and sounds. Most settings are self explanatory but note the following:

- Both time signature and tempo have a set of associated presets. To add or remove presets use the buttons at the bottom of each popup.
- The tempo can be adjusted by selecting a preset from the drop down, using the up/down arrow keys while the tempo indicator has focus, using the +/- buttons on the toolbar, or the slider.
- The last two controls enable the metronome sounds and adjust their volume. (See below for configuring metronome sounds).
- The current tempo and time signature are saved in the song file and in Cantabile Performer can also be controlled by [states](#).
- The tempo presets, time signature presets and metronome sound settings are saved globally (ie: not per song).

## Starting and Stopping the Metronome

The metronome can be started, stopped and paused using the transport controls on the main toolbar. The status panel will update to show the current transport position, timing information and current beat.



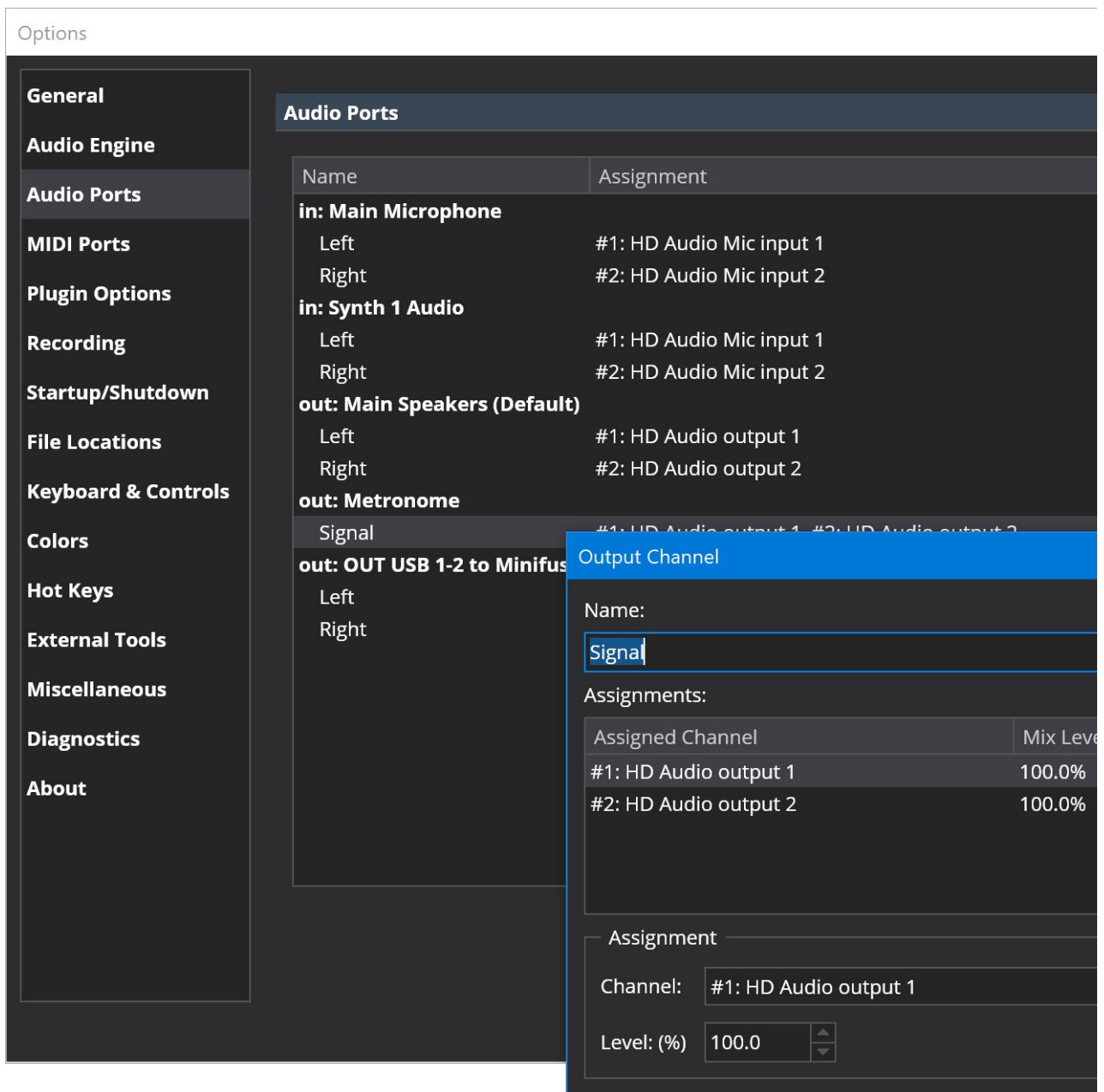
## Synchronizing to an External MIDI Clock Source

Cantabile can also synchronize it's timing to an external MIDI clock source and act as a MIDI clock master to other devices. For more information about this, see [MIDI Clock](#).

## Routing Metronome Sounds

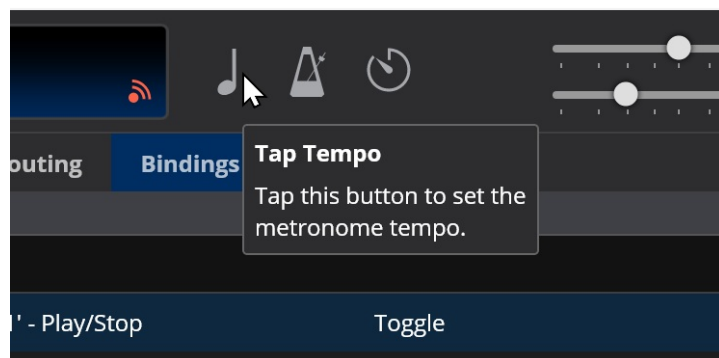
In order to hear sounds from the metronome you must configure an audio port named "Metronome". All metronome sounds are sent to this port. If it's not present, or not connected to active speaker channels you won't hear the metronome sounds.

Cantabile will automatically create an audio port for the metronome however you need to make sure the channel is mapped to at least one audio driver channel. In the following example the metronome signal has been mapped to both main speakers (HD Audio Output 1 and 2).

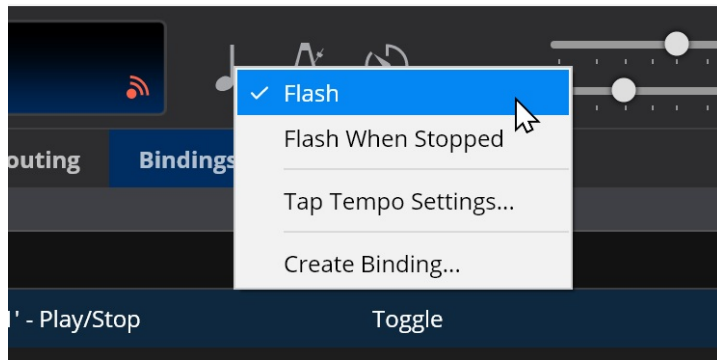


## Tap Tempo and Tempo Flash Indicators

The metronome tempo can also be set by tapping the "Tap Tempo" button on the main toolbar (or the equivalent button on the [controller bar](#)):



You can also configure the Tap Tempo button to flash in time with the current tempo:



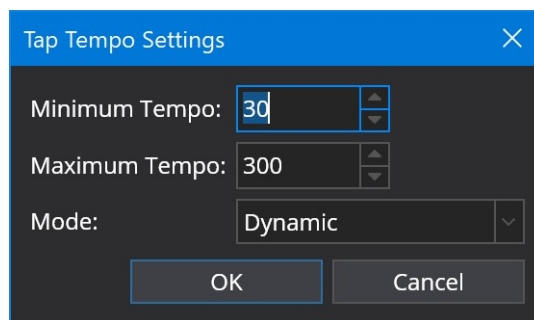
Right click on the Tap Tempo button and choose from the following:

- Flash - flashes in time with the current master transport when it's playing.
- Flash When Stopped - flashes in time with the master transport when playing but also flashes the current tempo even when the transport is stopped (but it's not synchronized with any timing information used internally by Cantabile).

The second option is useful if you don't actively use the metronome for timing but need a visual indicator of a specific tempo.

## Tap Tempo Settings

By right clicking on the tap tempo button on the main toolbar, you can access various options that control how tap tempo behaviour:



The minimum and maximum tempo settings constrain the range of tempos that will be detected. The Mode controls how the tempo is calculated:

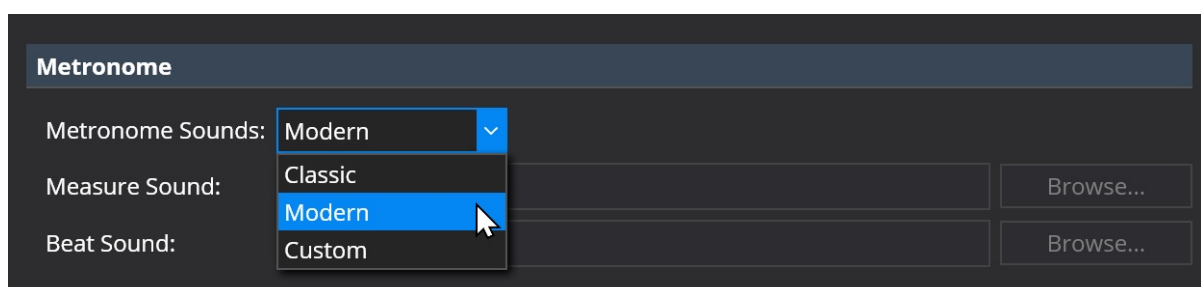
- Dynamic - best for tracking dynamically changing tempo in real-time
- Moving Average - calculates an average tempo of the last 16 taps
- Overall Average - calculates an average tempo for every tap since the tempo detection was last reset. Best for calculating an accurate tempo over a long period of time.

The tempo detector resets itself after 2 seconds of receiving no taps.

## Changing Metronome Sounds

There are two built in sound sets for the metronome - Classic and Modern, but you can also select custom sounds if you prefer.

You'll need two sounds files one for the first beat of the bar/measure and one for the other beats. These files must be wave files and only the first two seconds of each is used.



These settings can be found on the General options page.

## Disabling Measure Sounds

By default the metronome sounds use a different sound for the first beat of each measure. You can disable this sound and use the same sound for all beats:

1. Right click on the button to enable metronome sounds
2. Choosing "Enable Measure Sounds" to toggle the measure sounds on/off.

## About Metronome Bindings

Like most other settings in Cantabile the metronome can be controlled remotely via bindings. There are however a few nuances about the metronome bindings that you should be aware of.

The metronome tempo ranges from 30 to 300 bpm. Normal MIDI continuous controllers (CCs) however have a range of 0 - 127. The tempo binding supports setting a target range so you can set how to map the CC value onto a tempo.

The metronome time signature can be bound either by selecting a preset (the drop down time signatures in the metronome bar) or by direct bindings to the time signature numerator and denominator.

The numerator ranges from 2 to 12.

The denominator supports value 1, 2, 4, 8, 16 and 32. Other values will be ignored.

The denominator ( $2^n$ ) binding accepts values where the denominator is calculated as  $2^n$  and n is a value from 0 to 5. This matches the MIDI methods for specifying time signature denominators:

- 0 = /1
- 1 = /2
- 2 = /4
- 3 = /8
- 4 = /16
- 5 = /32

When creating bindings to the time signature the range should be set according to how you're trying to control the tempo. If you're binding a knob or slider to the time signature you should map the full range of the controller to the full range of the binding target.

eg: 0..127 → 1..12

This will allow moving the slider from one end to the other to select the full range of values.

If you're trying to programatically control these values you'll probably want to change the binding's range mapping to a 1:1 mapping

eg:

- Numerator: 1..12 → 1..12
- Denominator: 2..32 → 2..32
- Denominator ( $2^n$ ): 0..5 → 0..5

This will let you send explicit values to directly set the metronome settings.

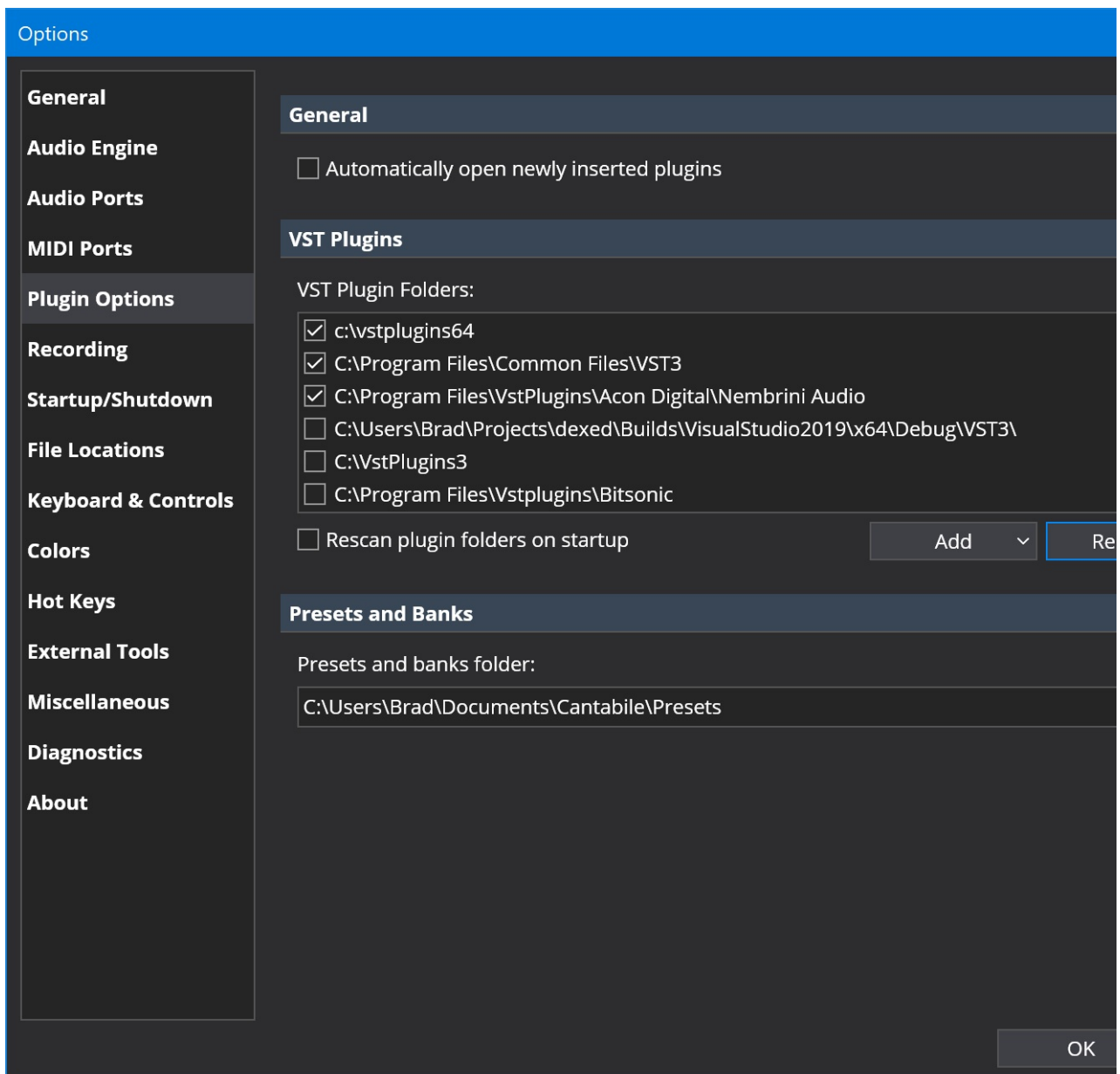
## Managing Plugins

Cantabile provides various tools for managing and organizing plugins.

### Setting the VST Plugin Path

Cantabile needs to know where you've installed your VST plugins. This is done in Options → Plugin Options:





When you change the VST path, Cantabile will scan the specified folders to locate information about all of your installed plugins. The discovered plugins will then appear in the Insert Plugin window.

Note:

- in the above screen shot you can also specify a "Presets" folder. Any plugin banks or programs (eg: .fxb, .fxp, .cantabileBank, .cantabileProgram etc...) files saved to this folder will also appear in the Insert Plugin window.
- you can specify multiple VST folders by separating each with a semicolon (;) character.

## Scanning Plugin Folders

If you install new plugins to your VST path, a new plugin scan will need to be run before those plugins are discovered. If you turn on the option "Rescan plugin folders on startup" (see above screen shot) these new plugins will be automatically discovered when you restart Cantabile.

To discover new plugins that are installed while Cantabile is running you can manually invoke a plugin scan by choosing "Scan Plugin Folders" from the Tools menu.

There are two plugin scanning modes:

- Quick - looks for new and removed plugins, doesn't re-scan plugins that haven't changed
- Full - all currently known plugin information is discarded and every plugin is re-scanned.

## Inserting a Plugin

Once your plugin folder(s) have been scanned all the found plugins will be shown in the Insert Plugin window (Insert menu → Plugin...).

The Insert Plugin window has various ways of browsing and finding plugins:

- Type the first few letters of the plugin and the list will be filtered to only show those that match what you've typed.
- Use the categories in the left panel to browse for plugins of a particular type.
- Use the "By Folder" category to browse plugins as they're organized on disk.
- Use the "By Tags" category to browse plugins organized with your own tags (see below).

Once you've found the plugin you're after either double click it, or select it and press OK to insert it.

## Plugin Information

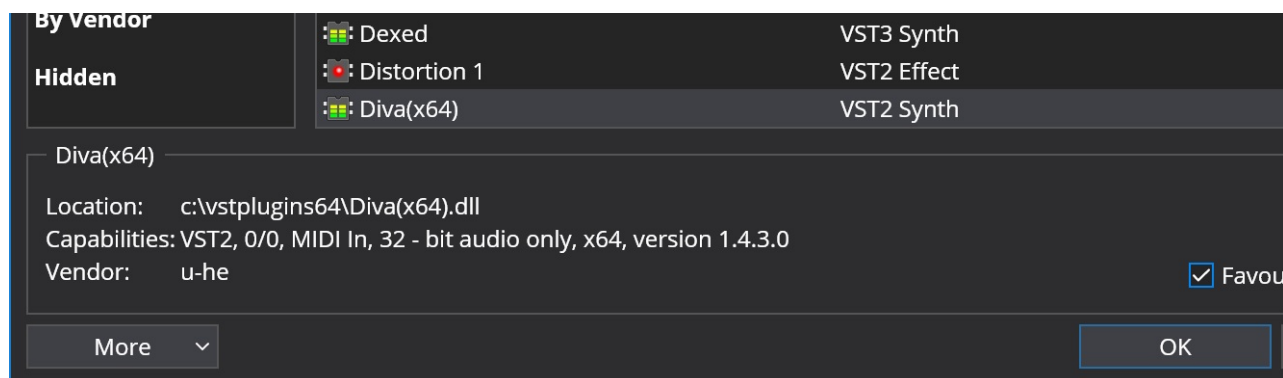
When a single plugin is selected in the plugin list, information about that plugin will be shown in the panel below the list, including:

- The full path the plugin on disk
- The plugins capabilities such as the number of audio inputs/outputs, whether it supports MIDI input output, audio precision support etc...
- The plugin vendor ie: who developed the plugin

## Favourite Plugins

You can mark plugins as being a Favourite to have it appear in the Favourites category:

- To mark a plugin as a favourite, select it in the list and turn on the "Favourite" checkbox.



## Hidden Plugins

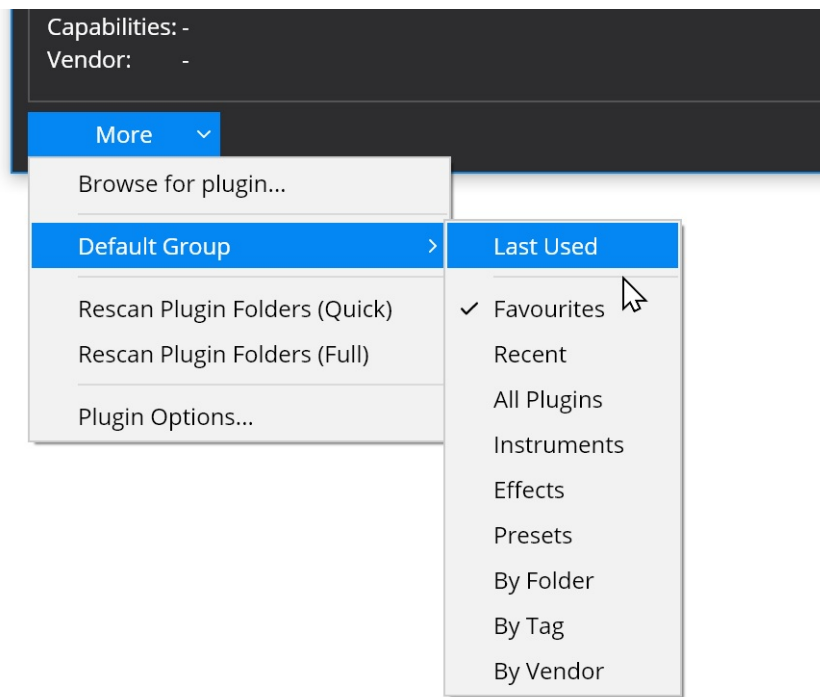
You can also mark plugins as hidden in which case they will only appear in the "Hidden" category and will be excluded from all other categories and search results.

To unhide a plugin, find it in the Hidden category and remove the checkmark from the Hidden option.

## Default Plugin Category

You can choose which plugin category is selected when you first bring up the Insert Plugin window via the "More" button drop down.

eg: suppose you want the Insert Plugin window to always start on the Favourites group:



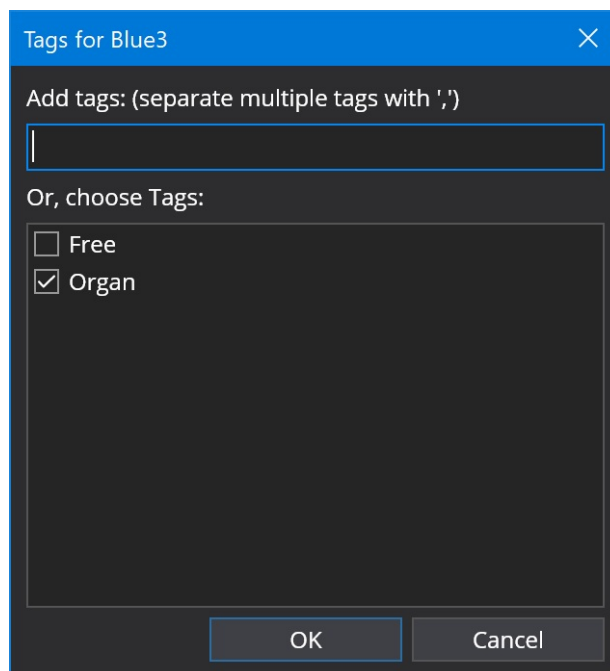
## Plugin Tags

If you have a large library of plugins you might like to organize your plugins using "tags" - a list of words that better describe the type of plugin. eg: you might like to tag Kontakt as a "Sampler" and VB3 as an "Organ".

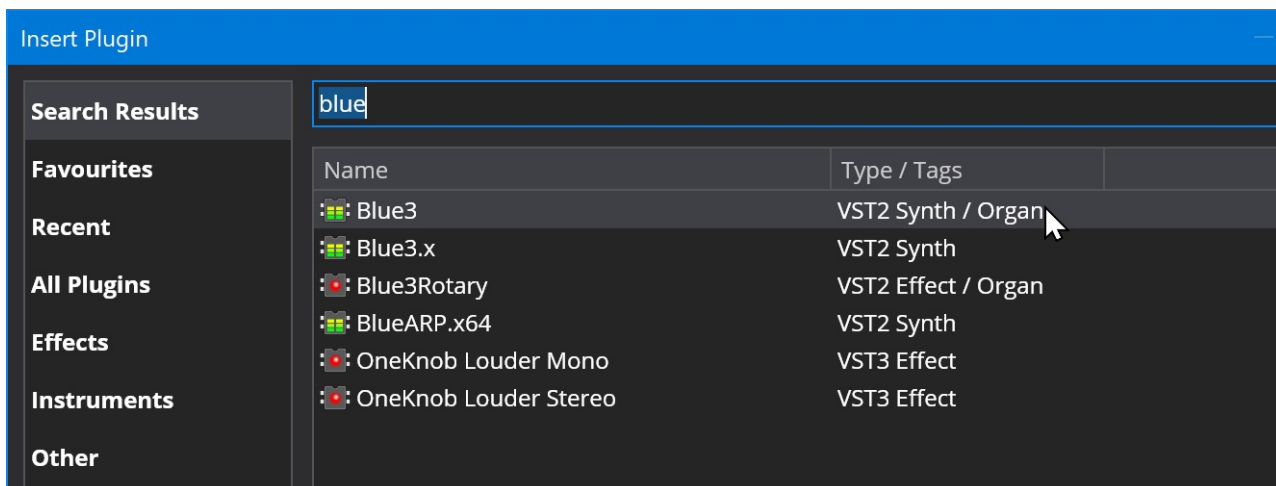
To set a plugin's tags:

1. Select the plugin in the Insert Plugin window
2. Right click on it and choose "Edit Tags", or Ctrl+T

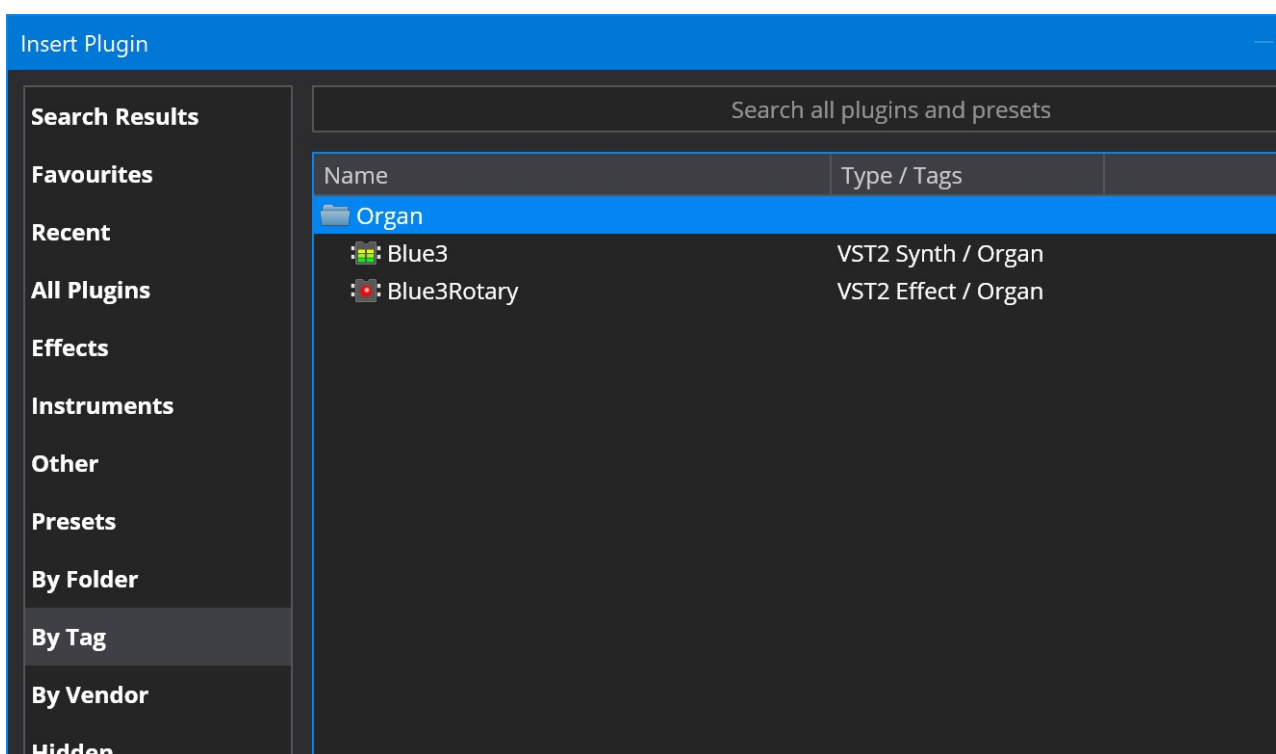
You'll be presented with the Edit Plugin Tags window and the first time you bring it up it will be empty. Just type the names of the tags you'd like to assign to this plugin and press OK:



Back in the Insert Plugin window, you'll notice the second column now shows the tags:



and if you select the "By Tag" group on the left, you'll be able to browse all plugins with particular tags:



Also note:

- To remove a tag from a plugin, bring up the Edit Tags window and remove the check mark for that tag.
- You can set the tags for multiple plugins at once - just select all the relevant plugins before bringing up the Edit Tags window.

## User Information Storage

All user entered information about plugins (favourite plugins, hidden plugins and plugin tags) are stored in the file "plugins.user.json" which can be found in the settings folder (Tools menu → Open Settings Folder).

Please be sure to back up this file so you don't lose these settings. You can also copy this file between configurations to have the same tags across multiple configurations.

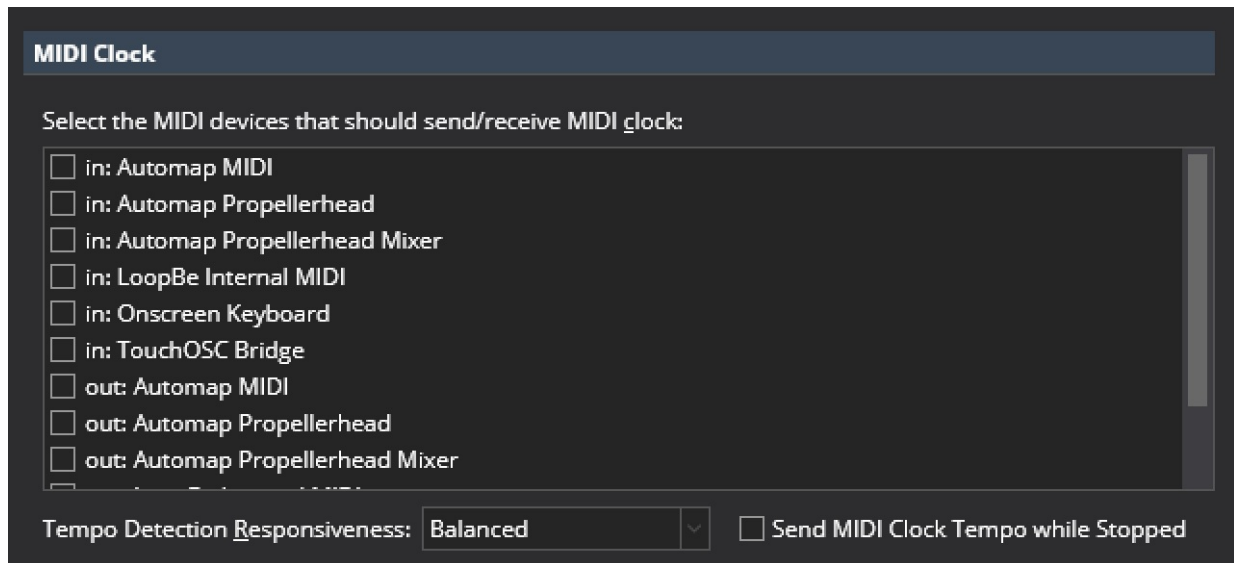
## MIDI Clock Synchronization

*Cantabile Solo and Cantabile Performer Only.*

MIDI clock allows Cantabile to synchronize it's timing with and external MIDI clock source, or to acts as a master to which other devices can synchronize their timing.

## Setting up MIDI Clock Synchronization

Before MIDI clock synchronization will work you need to select which MIDI ports Cantabile should use. Selecting an input port allows Cantabile to synchronize to incoming MIDI clock events. Selecting an output port causes Cantabile to send MIDI clock events to that port.



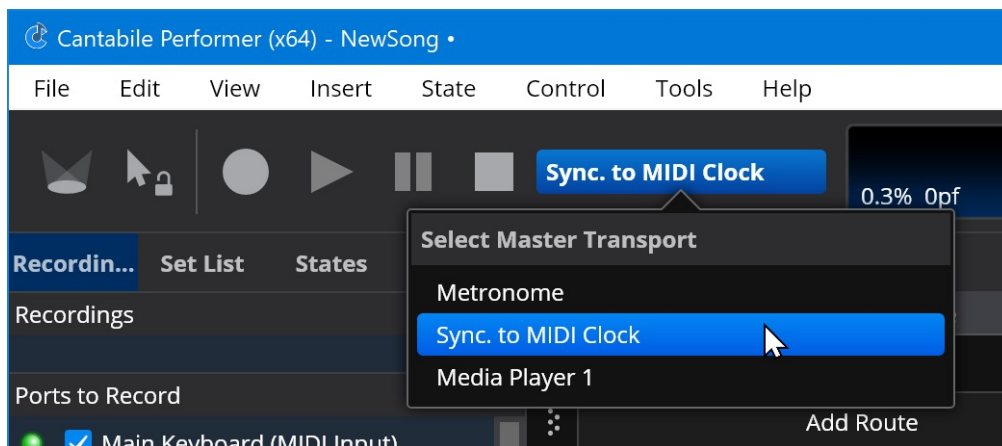
Other options include:

- Tempo Detection Responsiveness - how responsive Cantabile should be when detecting incoming MIDI clock tempo. Choose between:
  - Fast (but unstable)
  - Balanced
  - Slow (but stable)
- Send MIDI Clock Tempo while Stopped - normally Cantabile doesn't send MIDI Clock tempo ticks while the transport is stopped. Enable this option to have tempo transmitted while the transport is stopped.

Note: When "Send MIDI Clock Tempo while Stopped" is enabled there will be a small amount of tempo jitter when switching between playing and stopped modes as Cantabile re-synchronizes to it's own master transport.

## Synchronizing to and External MIDI Clock

To synchronize to an external MIDI clock, choose "Sync. to MIDI Clock" from the master transport drop down:



When enabled, you'll notice that the options to select tempo are disabled, as are the transport controls (Play, pause etc...) since these functions are now controlled by the external MIDI clock device.

If you enable MIDI clock events on multiple MIDI input devices, Cantabile will monitor all selected devices for clock events but only respond to one at a time. While one device has an actively playing transport, other device's clock events will be ignored.

## Synchronizing External Devices to Cantabile

To synchronize external devices to Cantabile's MIDI clock all you need to do is select one or more devices in options and Cantabile will automatically send MIDI clock events. Refer to the documentation of your other device for information on how to configure it to synchronize to an external MIDI clock device.

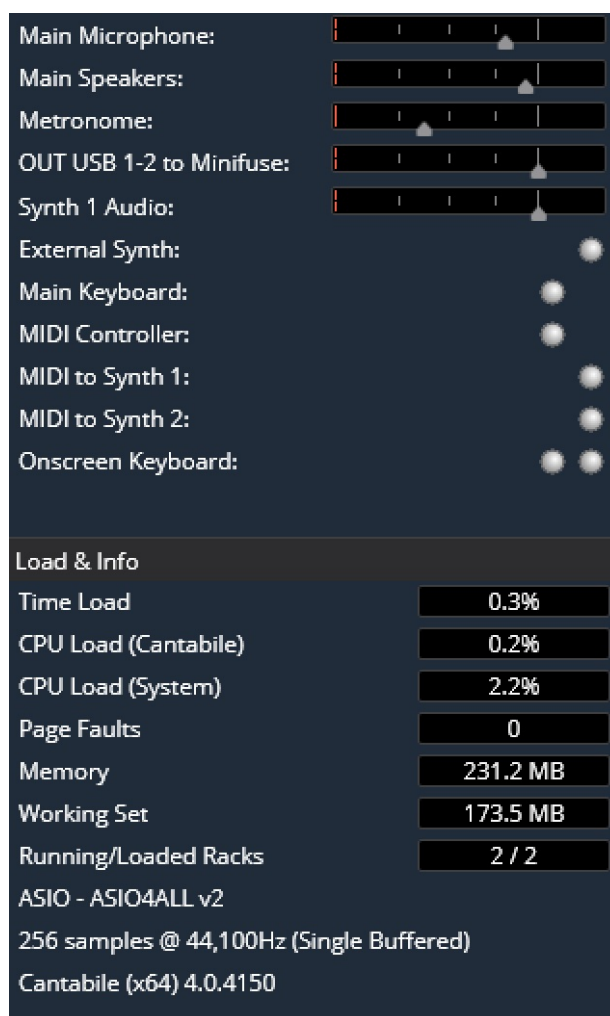
If multiple MIDI devices are selected for MIDI clock output, clock events will be sent to all selected devices.

## MIDI Clock Passthrough

When Cantabile is configured to synchronize to an external clock source and configured to send MIDI clock events, the incoming clock events are passed directly to the target devices (as opposed to Cantabile deriving MIDI clock events from the incoming events).

## Monitor Panel

The monitor panel displays audio level and MIDI activity indicators for all of the currently active environment ports. It also displays important performance metrics of the audio engine.



To show the monitor panel, select "View" menu → "Side Panel" → "Monitor", or click the monitor tab on the side panel.

## Ports Display

The list of displayed ports automatically updates to include all ports enabled in Options → Audio Ports and Options → MIDI ports.

Unlike other audio level meters which are limited in how many channels they can display, the level meters in the monitor panel expand to show all channels.

These audio level meters also incorporate a gain level slider - click on the level meter to adjust gain levels on audio ports.

## Load and Info Display

The bottom half of the monitor panel displays the following performance metrics:

- Time Load - maximum processing load as a percentage of the audio cycle time over the previous 1 second (this is the most important metric and indicates how close to audio dropouts and glitches the current load is)
- CPU Load (Cantabile) - CPU Load caused by Cantabile and all loaded plugins
- CPU Load (System) - The system-wide CPU load
- Page Faults - the number of hard page faults in the previous 1 second
- Memory - total reserved memory used by the process
- Working Set - total amount of memory actually committed to physical RAM
- Running/Loaded Racks - the number of currently running racks and the total number of loaded racks (may differ when [Set List Pre-loading](#) is in use)
- Audio driver type and name
- Audio driver buffer size and sampler rate
- Cantabile version and build number

See also [Understanding Cantabile's Performance Metrics](#).

## Morph and Randomize Tools

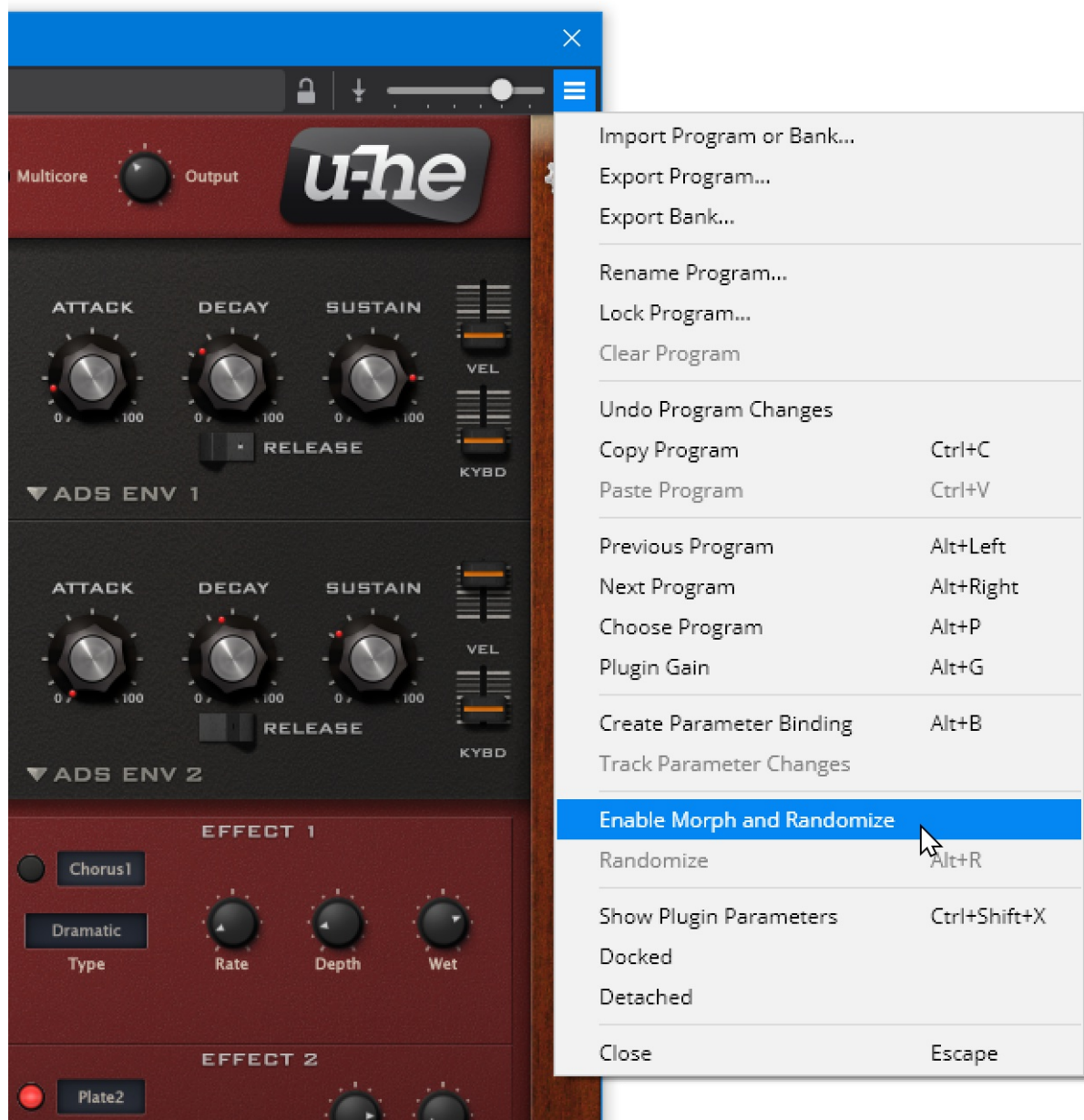
Cantabile includes tools for Morphing and Randomizing a plugin's parameters. This walkthrough explains how to use these features.

### Enabling Morph and Randomize

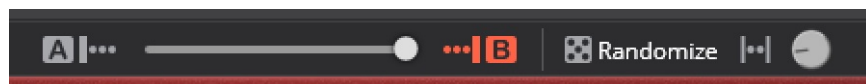
The morph and randomize settings for a plugin are stored on a per-plugin-program basis. That is, each of a plugin's programs has its own set of morph and randomize settings.

To enable morph and randomize for a particular program:

1. Open the plugin's editor or parameter editor by double clicking (or Alt+double clicking) the plugin slot in Cantabile's main window.
2. From the menu button in the top right hand corner, select "Enable Morph and Randomize"



3. You'll notice a new tool bar appear with the controls for morphing and randomizing the program's parameters.



4. If you switch between programs you'll notice that the morph and randomize tools can be enabled on some programs, disabled on others and for those programs where it's enabled each program has its own set of morph/randomize settings.

## Morphing

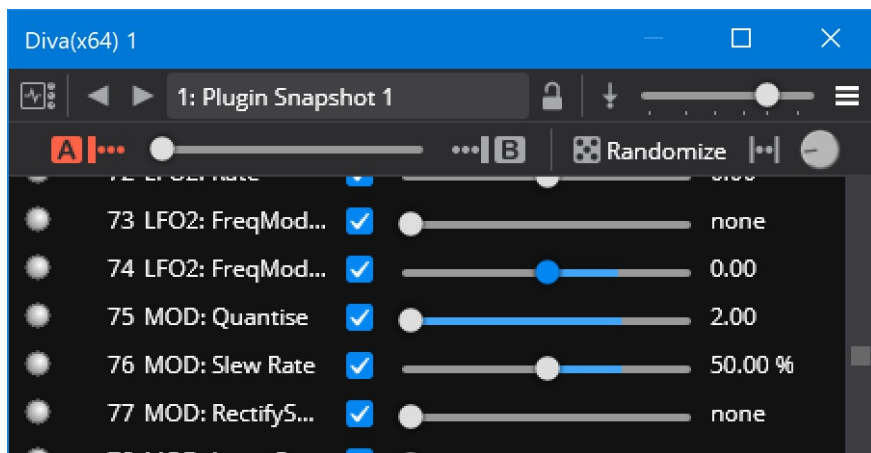
Morphing lets you configure two sets of plugin parameters and gradually move between them. Suppose for example you have to existing programs but you're after a sound part way in between you can load those two programs and "morph" between them to find the sound you're after.

The two sets of parameters used for morphing are called the "A" set and the "B" set. When you first turn on morphing, the both sets initialized to the plugins current parameter settings and the B set is selected.

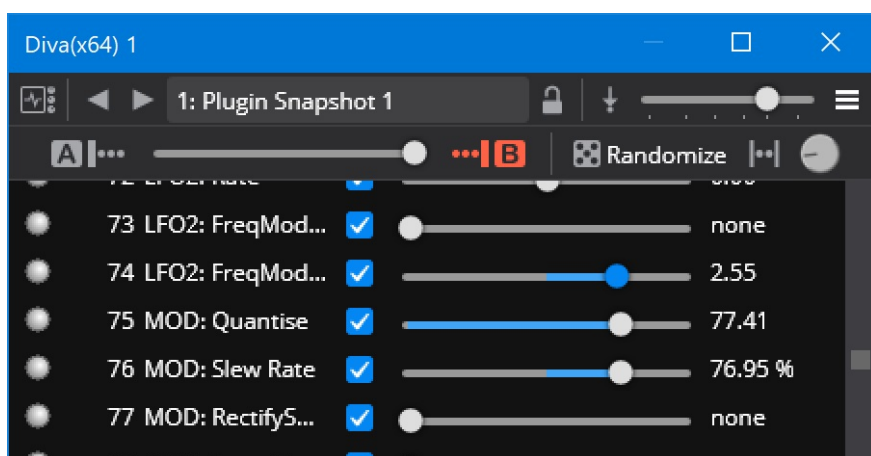
To make changes to the A and B sets click the matching button on the morph toolbar and edit the parameters. To morph between the parameter settings move the morph bar.



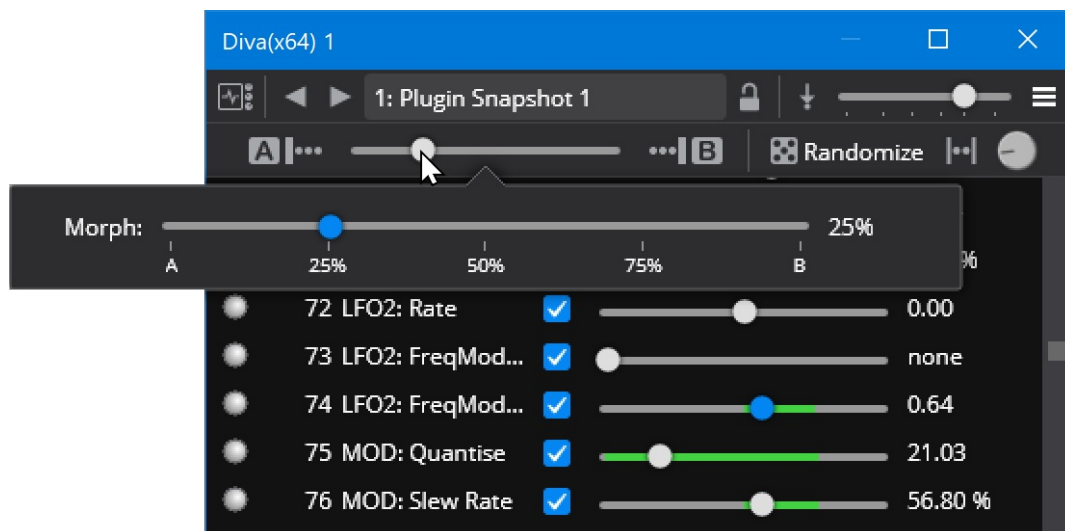
Here's a simple example... this is the "A" set:



and this is the "B" set:



and this is 25% morphed from A to B:



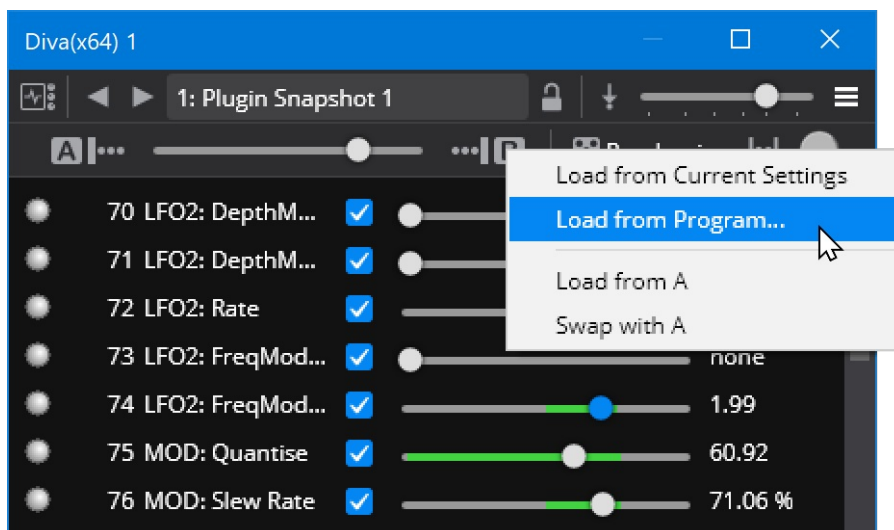
This example is using Cantabile's built-in parameter editor which shows the morph ranges for each parameter as highlights on the slider tracks. The same operation can be done using the plugin's GUI editor however the morph ranges aren't shown in that case.

Also note that the colour of morph ranges shown on the slider changes from green to blue:

- Blue - indicates you're editing the A/B parameter sets
- Green - indicates the parameters are partially morphed. Moving a slider on a blue range will cancel morphing that parameter (see inactive parameters below).

## Morphing Between Plugin Programs

To morph between two of a plugin's existing programs (aka Presets) you need to load the second preset into the "B" parameter set. To do this, right click on the "B" button and choose "Load from Program":



You'll be prompted for which program you want to load. You can perform a similar operation by right clicking on the "A" set.

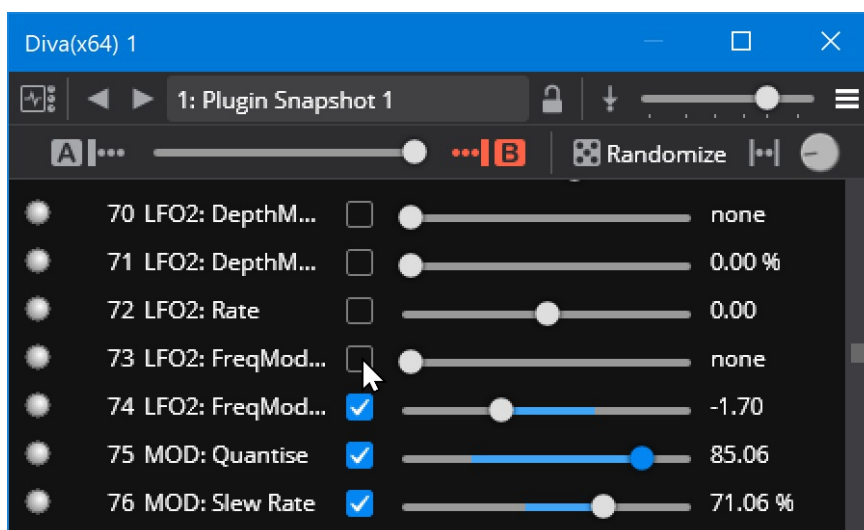
## Disabling or Cancelling Parameter Morphing

A parameter can be on one of three states for morphing:

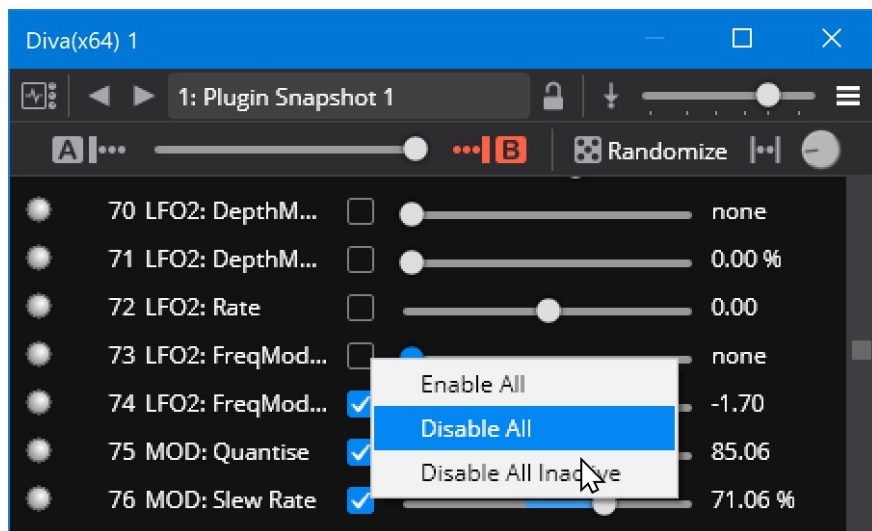
- Active - different values in the A and B parameter sets
- Inactive - enabled, but having the same values in the A and B sets
- Disabled - excluded from all morphing operations.

Inactive and disabled parameter behave similarly except that if you edit an inactive parameter in the A or B sets it will automatically become active.

- To deactivate an active parameter move the morph slider between the A and B points and edit the parameter.
- To activate an inactive parameter adjust it to have different values in the A and B sets
- To enable or disable a parameter use the check box in Cantabile's parameter editor.



You can also enable all, disable all or disable all inactive parameters by right clicking the parameter editor:



(Once you're happy with a morph set, disabling all inactive parameters can reduce the size of the saved song/rack file by not saving redundant sets of parameters).

## Randomizing Parameters

The randomize tools can be found on the right hand side of the morph randomize toolbar and consist of the following settings:

- The Randomize button - randomizes the parameters according to current settings
- Constrained Randomize setting - limits randomization to the current A/B parameter ranges
- Randomize Amount - how much to randomize settings by

Using the randomize tool is fairly straight forward - adjust settings and click the Randomize button.

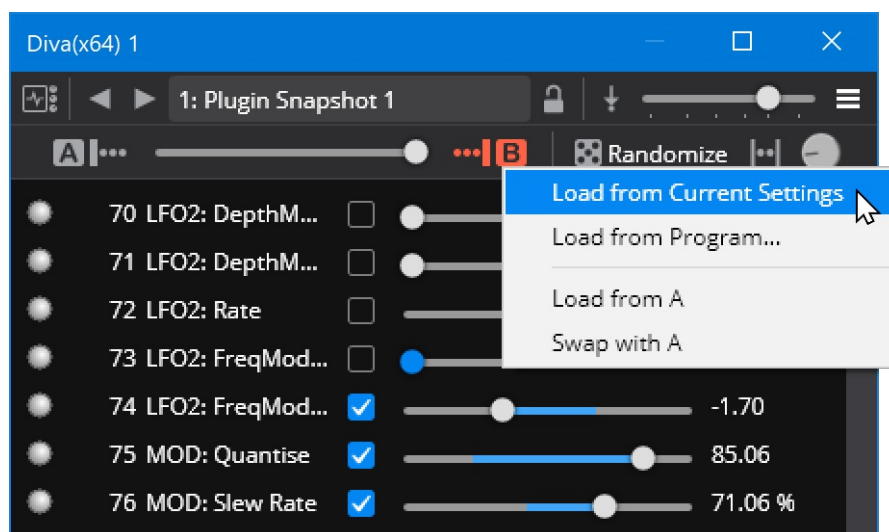
## Using the Randomize and Morph Tools Together

You can use the morph and randomize tools together to come up with unique new sounds.

When the constrain randomization option is turned off, clicking the randomize button while either the A or B parameter set is selected will directly update the A and B parameter sets.

When the constrain option is turned on - it doesn't make sense to automatically update the parameter sets and this will affect the constrained range. In this case you can:

1. Setup your range constraints on the A and B parameter sets
2. Turn on constrain option
3. Click the Randomize button as required
4. To apply the current randomize settings to the A or B parameter sets - right click that button and choose "Load from Current Setting":



## Binding Support

In Cantabile Solo and Performer you can morph and randomize via [Bindings](#). The following binding points are available on the plugin:

- Morph Position
- Randomize Limit
- Constrained Randomization
- Randomize

For example this binding will use CC 16 on the Main Keyboard port to control the morph setting for the "Diva(x64) 1" plugin:

The screenshot shows the 'Binding - NewSong - General - #1' window. It is divided into three main sections: Source, Target, and Mapping. The Source section includes a 'Learn...' button, and dropdowns for Object (MIDI Ports), Point (Main Keyboard (in)), Event (Controller), Controller (16), Ch (Omni), and Routing Mode (Continue). The Target section has dropdowns for Object (Plugin: 'Diva(x64) 1') and Point (Morph Position). The Mapping section includes fields for Mapper, Prevent Jumps, Source Range, Target Range, Out of Range, Curve Kind, and Curve. At the bottom, there are checkboxes for 'Enabled' (checked) and 'Bi-directional' (unchecked), along with 'Timing...' and 'Test' buttons.

## Network Server

*Cantabile Performer Only*

Cantabile includes a built-in network/web-server that allows remote control of Cantabile from other machines and supports building custom integrations between Cantabile and other systems.

### Enabling the Network Server

By default the network server is disabled, to enable it go to Options → Miscellaneous and turn on "Enable Network Server":

The screenshot shows the 'Network Server' configuration window. It has a title bar 'Network Server'. Below it, there is a checkbox 'Enable Network Server' which is checked. To the right of this checkbox is a link 'Configuration Guide'. Below the checkbox is a label 'URL Prefixes:' followed by a text input field containing 'http://+:35007/'.

Once enabled, in any web browser on the same local network, enter the IP address of the machine running Cantabile and use the port number 35007.

eg: suppose Cantabile is running on a machine with IP address 10.1.1.20 then in a web browser on any other machine on the network, enter `http://10.1.1.20:35007` to bring up the Cantabile web interface.

## Security

The network server isn't password protected nor authenticated. Currently the only way to restrict access to the network server is via Windows Firewall and/or by isolating to a local network.

## Firewall Settings

Cantabile's installer program creates the appropriate firewall rules and URL permissions to allow the web server to run.

If you want to run the network server on a different port, you'll need to:

1. Open the appropriate port in the Windows firewall
2. Set the appropriate URL access permissions

For details on how to do this, refer to the `configureNetwork.bat` file in Cantabile's installation directory.

## Developer Guide

For developers interested in working with Cantabile's network server, see the [Network Developer Guide](#).

## Offline Rendering

*Cantabile Performer and Solo Only.*

Cantabile's offline renderer can be used to process a MIDI or audio file through a set of racks and plugins and save the results to a wave file. This rendering process is done offline – ie: not in real-time.

The main advantages of offline rendering are:

- When using plugins that can process faster than real-time, the rendering can be done more quickly.
- When using plugins that can work in higher quality but more CPU intensive modes, those modes can be used to render high quality sounds without audio drop outs.

Note that the offline renderer is not compatible with most hardware based plugins and other plugins that don't handle faster than real-time processing.

## Using the Offline Renderer

To use the offline renderer:

1. Load a song with a media player configured to play the audio or MIDI file you want to process
2. Set the media player as the [master transport](#)
3. Optionally, in the timeline panel, select the range of the file you want to render
4. From the File menu, choose the **Offline Render** command.
5. Configure the render settings as explained below
6. Press OK to process the audio and produce a .wav file.

Offline Render

Output File:

\$(SongTitle) (Rendered)

▶

Output Folder:

\$(SongFolder)

▶

C:\Users\Brad\Documents\Cantabile\LiveLoop (Rendered).wav

Duration:

00:00:00.000

Sample Rate:

48,000Hz

▼

Sample Format:

32-bit floating point

▼

Lead-in:

1.00

▲▼

Lead-out:

1.00

▲▼

☒ Normalize Output Level:

-3.0 dB

▲▼

☒ Disable Loop Ranges

Record Ports:

☒ Main Speakers  
☐ Metronome  
☐ Out71

2 channels

OK

Cancel

## Settings

### Output File

The name of the file to be generated. Use variables to configure the name based on the current song or media file name.

### Output Folder

The folder to save the rendered file. Variables can be used in this field too.

### Duration

This field is only enabled and available if the master transport is not a media file and lets you render for a specified time period.

### Sample Rate

The sample rate of the rendered audio file

### Sample Format

The sample precision and format of the rendered audio file

### Lead-in

An optional period of silence (in seconds) to insert at the start of the rendered audio file

### Lead-out

An optional period to continue processing after the end of the rendered media file. Set this to a value high enough to capture any trailing sounds produced by instruments and effects after the original media file has stopped playing.

### Normalize Output Level

When enabled, post-processes the audio file to normalize it to a specified output level.

### Disable Loop Ranges

When enabled, causes loop ranges in the media file to be ignored.

### Record Ports

The set of audio ports to be recorded to the output file. Each port will be recorded as a consecutive set of parallel audio channels in the output file.

## Notes

Please note the following about offline rendering:

- During offline processing all bindings are disabled. If you're using bindings to trigger state changes or other effects during the playback of a media file, these bindings won't be triggered during offline rendering
- During offline processing all physical audio and MIDI ports are disabled.
- During offline processing the network API (including Web UI and Stream Deck plugin) are all disabled.
- The offline renderer uses the selected play back range and loop count of the media player. To render and entire media file with out the play range or looping deselect these options in the timeline panel before invoking the offline render command.

## Onscreen Keyboard

Cantabile includes an on-screen keyboard that can be used to either inject MIDI events into a song (eg: to play an

instrument) or to show the notes coming from a MIDI source.



### Setting up the Onscreen Keyboard

The on-screen keyboard works just like any other MIDI device in that it needs to be configured via a MIDI port. Normally Cantabile will automatically create a both an input and output port and map it to the on-screen keyboard however you can re-configure if necessary:

Options

General

Audio Engine

Audio Ports

MIDI Ports

Plugin Options

Recording

Startup/Shutdown

File Locations

Keyboard & Controls

Colors

Hot Keys

External Tools

Miscellaneous

Diagnostics

About

MIDI Ports

Name	Settings
<input checked="" type="checkbox"/> in: Main Keyboard (Default)	Onscreen Keyboard
<input checked="" type="checkbox"/> in: MIDI Controller	Onscreen Keyboard
<input checked="" type="checkbox"/> in: Onscreen Keyboard	Onscreen Keyboard
<input checked="" type="checkbox"/> out: External Synth	loopMIDI Port 1
<input checked="" type="checkbox"/> out: MIDI to Synth 1	Onscreen Keyboard
<input checked="" type="checkbox"/> out: MIDI to Synth 2	Onscreen Keyboard
<input checked="" type="checkbox"/> out: Onscreen Keyboard	Onscreen Keyboard

MIDI Clock

Select the MIDI devices that should send/receive MIDI

☐ in: Automap MIDI

☐ in: Automap Propellerhead

☐ in: Automap Propellerhead Mixer

☐ in: LoopBe Internal MIDI

☐ in: Onscreen Keyboard

☐ in: TouchOSC Bridge

☐ out: Automap MIDI

☐ out: Automap Propellerhead

☐ out: Automap Propellerhead Mixer

Tempo Detection Responsiveness: 

Balanced

MIDI Input Port

Name:

Onscreen Keyboard

Aliases: (separate with commas)

Assignments:

☐ Automap MIDI

☐ Automap Propellerhead

☐ Automap Propellerhead Mixer

☐ LoopBe Internal MIDI

☒ Onscreen Keyboard

☐ TouchOSC Bridge

Listen on MIDI &Channels:

None

Omni

1

2

3

4

5

MIDI Filters...

MIDI Filters passing through

☒ Enabled

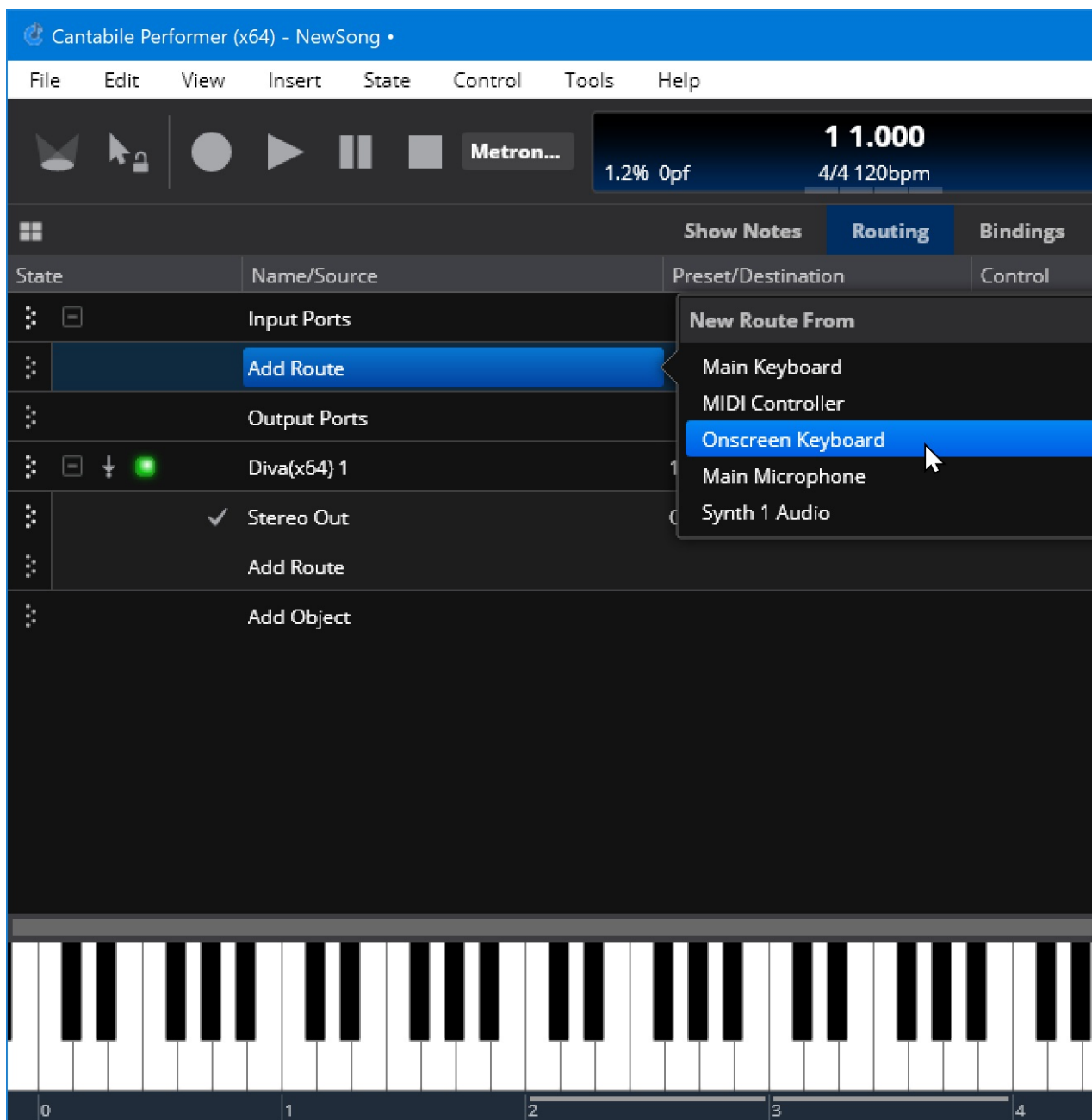
☐ Default Port

☒ Show in Monitor Panel

☒ Include in Master Indicator

☐ Route MIDI Clock

Once the appropriate MIDI ports are configured you can create routes to and from it just like you would any other MIDI device:



## Showing and Hiding the Keyboard

To show the onscreen keyboard:

- Choose *On-screen Keyboard* from the *View* menu, or
- Press *Ctrl+K*.

In either case if the keyboard is already visible it will be activated so it can be played using your PC's keyboard.

To hide the keyboard:

- Resize it (by dragging its top boarder) until it disappears.
- While the keyboard is active, press *Shift+Escape*.

To leave the keyboard visible, but move focus back to plugin list press *Escape*.

## Sending MIDI to the Onscreen Keyboard

Although normally used for playing notes the on-screen keyboard can also be used to show notes coming from other MIDI sources. To do this, create a route from an external device, the output of a plugin or any other MIDI source and connect it to the Onscreen Keyboard port (as shown in the screen shot above).



## Playing the Keyboard

The highlighted bars below the keyboard show the currently active octaves. You can play notes in these octaves using the keys on your PC keyboard:

- The lower octave can be played starting with the Z key, in a similar pattern to a standard piano style keyboard. eg: Z Key= C, S Key = C#, X Key= D etc...
- The other octave can be played starting on the Q key. eg: Q, 2, W, 3, E etc...

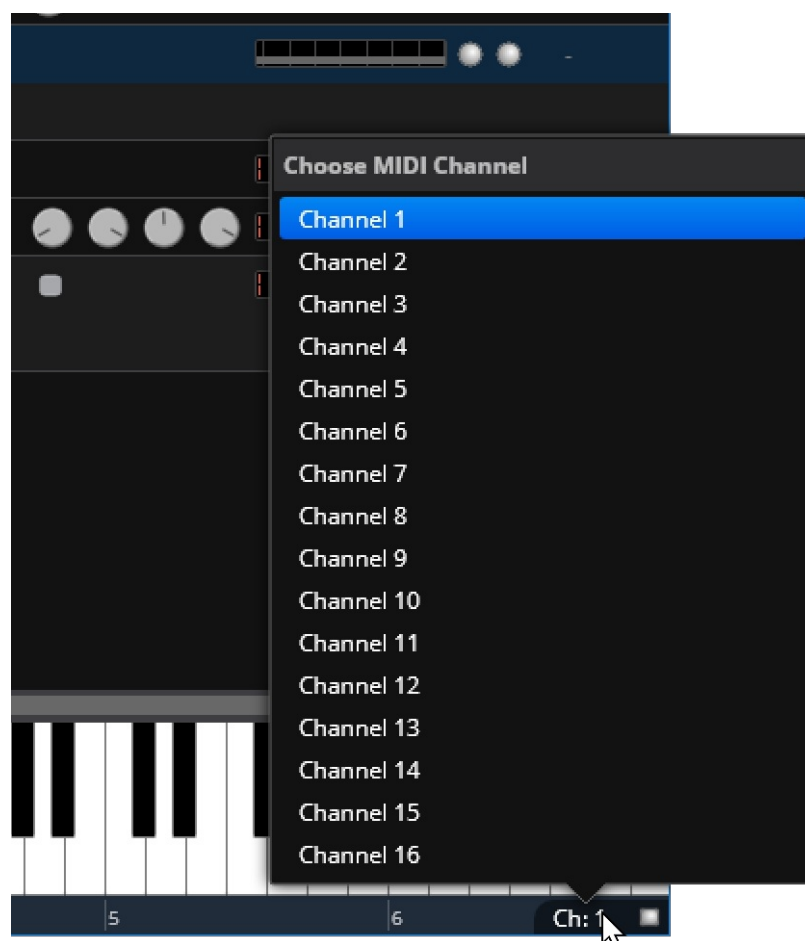
(Note you can actually go a little further than one octave in each range).

To change the active octave click the in the area below the keyboard or use the *Page Up/Page Down* or the *Up/Down* keys. You can also split the two octaves: hold the *Ctrl* key to change just the lower octave, or the *Shift* key to change the upper octave.

Playing the keyboard with the mouse is straight forward - just click the notes you want to play, or click and drag to play glissandos.

## MIDI Channel

The channel selector at the bottom right of the keyboard allows selection of the MIDI channel that the on-screen keyboard sends on.



## Velocity

When playing with the mouse, you can control the velocity of the played notes by the vertical position on the keyboard where you click. The area below the black keys plays at full velocity, but as you move closer to the top of the keyboard the notes will play more softly.

When playing using the PC keyboard, the notes are played using the last velocity played using the mouse.

## Sustain (Damper) Pedal

The sustain/damper pedal (MIDI CC 64) can be simulated by pressing the Shift key.

## Resizing and Scrolling

To resize the keyboard:

- Click the border at the top of the keyboard and drag up/down.

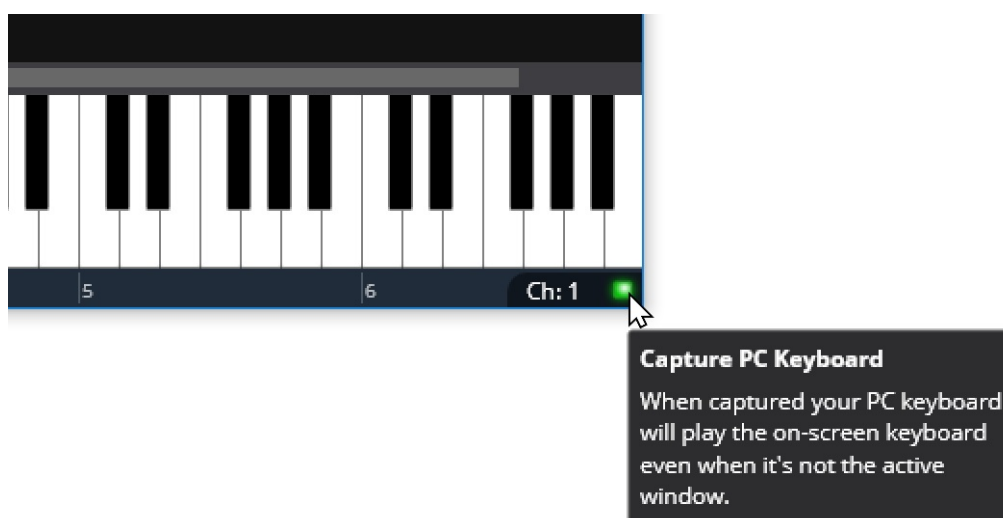
To scroll the keyboard:

- Click and drag in the area below the keys (ie: on the active bar indicators).
- Use the Left and Right arrow keys.

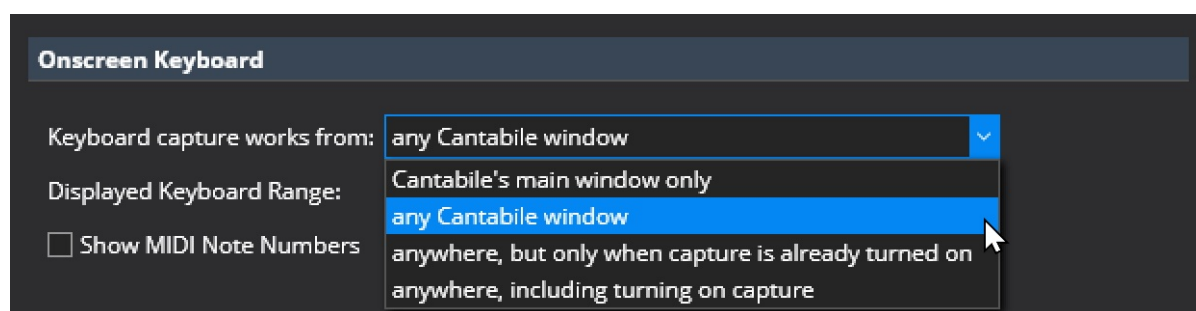
## PC Keyboard Capture

Keyboard capture lets you play the on-screen keyboard using your PC keyboard even when it doesn't have input focus.

To use the keyboard capture just hit the F12 key. You can then move focus anywhere in Cantabile's main window and playing the PC keyboard will continue play the on-screen keyboard. You can tell if capture is active by the green indicator in the bottom right hand corner next to the channel selector.



If you want the keyboard capture to work from other windows as well as Cantabile's main window, there are settings in Options → Keyboard and Controls → On-screen Keyboard that control this behaviour:

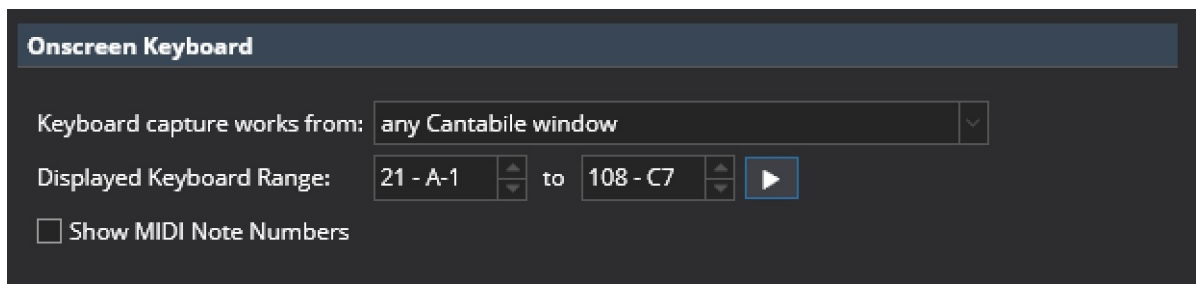


The last two options both allow you to play the on-screen keyboard when any window (even non-Cantabile windows) are active. The difference is the first option, capture must be enabled while Cantabile is active. With the second options the F12 key can be pressed while another application is active and it will enable the keyboard capture.

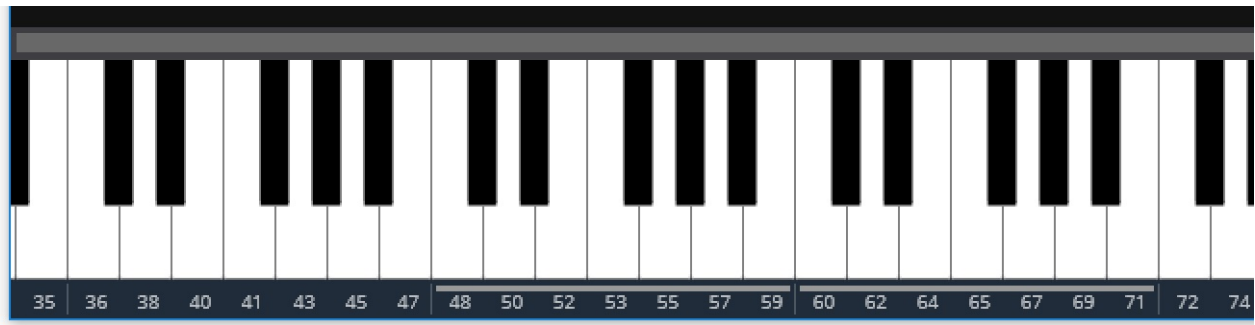
You can change the F12 key to a different key in Options → Hot Keys.

## Keyboard Range and MIDI Note Numbers

Usually you don't need the full 127 MIDI note key range on the on-screen keyboard. To adjust the available note range, go to Options → Keyboard and Controls → Onscreen Keyboard:

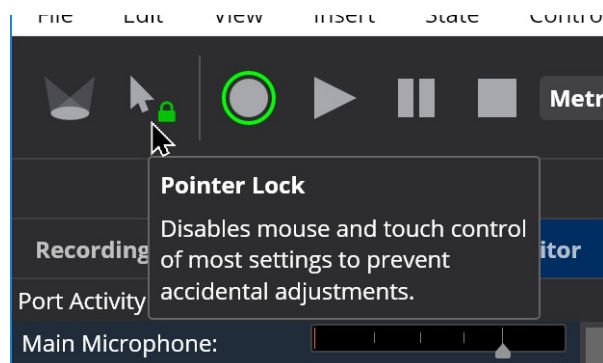


The "Show MIDI Note Numbers" option adds indicators to the keyboard to show the MIDI note numbers for each note (instead of their musical names):



## Pointer Lock

Cantabile's Pointer Lock feature is designed to prevent accidental adjustment of settings when using mouse and/or touch pad. When enabled, it locks down most sliders, knobs, preset selectors and buttons that aren't typically used during performance.

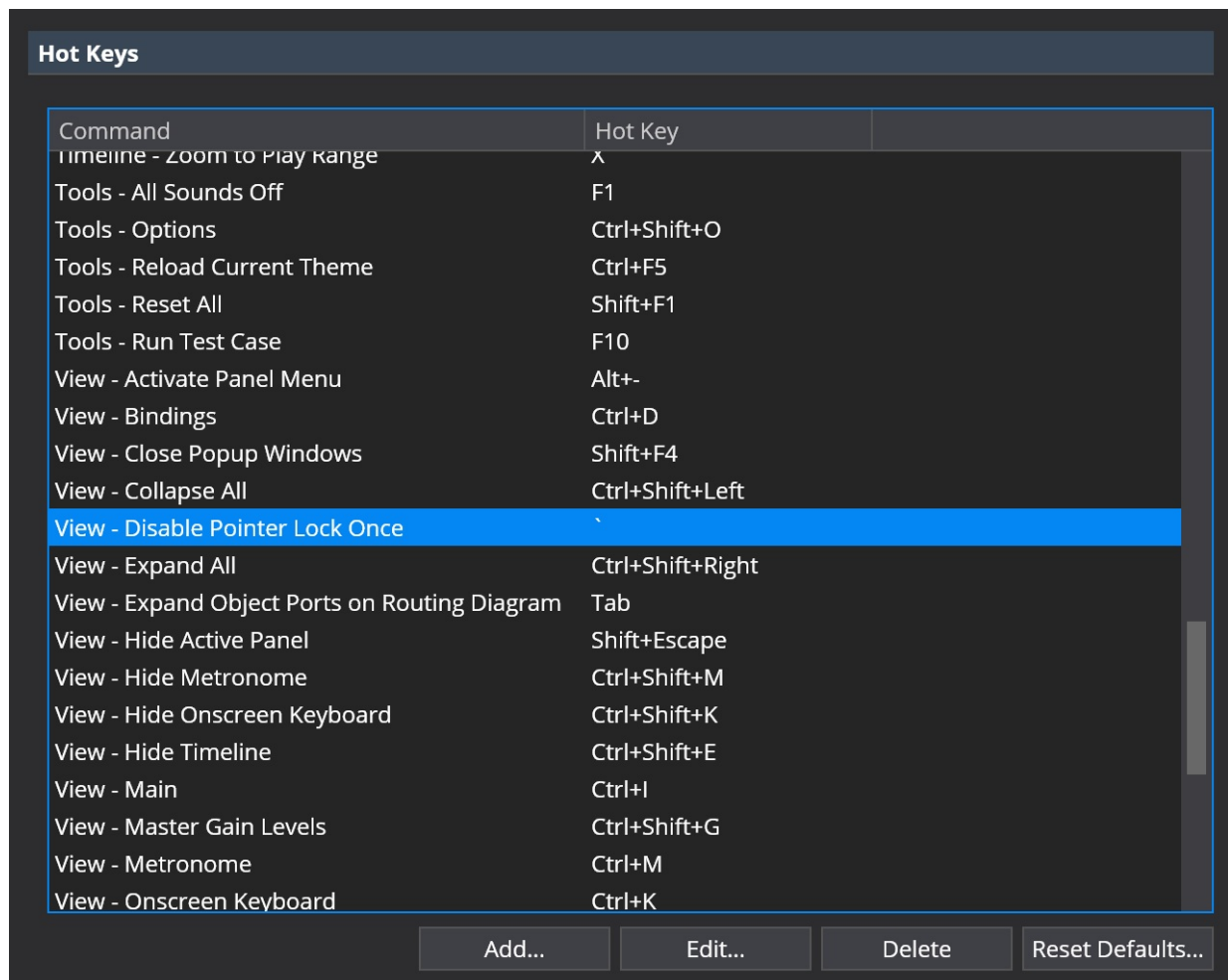


## Using Pointer Lock

- When pointer lock is enabled the lock icon in the toolbar button is shown as locked/green and clicking on any affected control will select the control but you won't be able to adjust its value.
- When trying to use a locked control, the toolbar button will flash to remind you it's locked.
- To make a quick one-time adjustment you can temporarily disable pointer lock by double clicking the toolbar button. This will let you make one adjustment before automatically relocking. (When in this mode the lock icon will remain green but appear unlocked).
- You can also temporarily disable pointer lock by pressing the backtick/tilda key. (see also below)
- Pointer lock doesn't prevent changing settings with the keyboard.

## Customizing the Key to Temporarily Disable Pointer Lock

If you're using a non-US keyboard the default key to temporarily disable pointer lock mightn't be the most convenient. To change the key, go to Options → Hot Keys and edit the hot key for the View - Disable Pointer Lock Once command:



## Pointer Lock and Live Mode

In Cantabile Performer the current pointer lock mode is captured by Live Mode. If you enable it in Live Mode, it will be automatically enabled every time you enter Live Mode.

## Plugin Editor

Most plugins have a user-interface that can be used to configure the plugin's settings. This user interface is displayed in Cantabile's plugin editor - a window that wraps the plugin's user interface and provides a couple of additional controls for working with the plugin.

In the following screen shot, the plugin's user interface comprises the majority of the window, while Cantabile's plugin editor toolbar can be seen at the top.

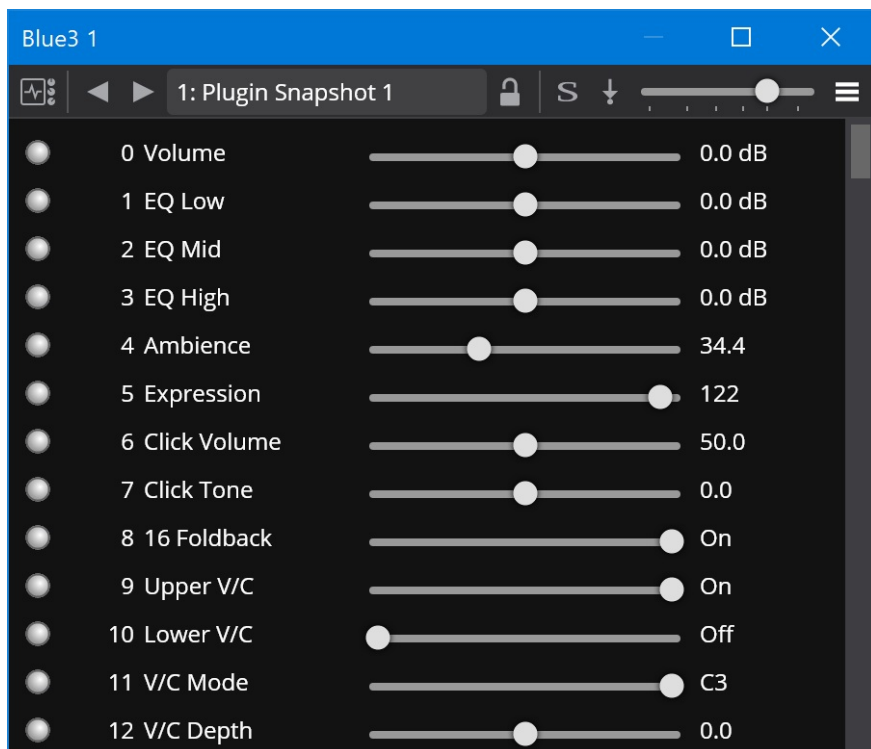


## Plugin Editor vs Parameters

Plugin settings can be adjusted using two different user interfaces:

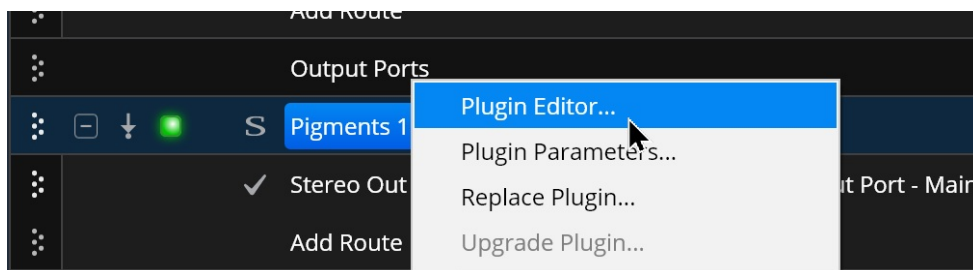
1. The plugin provided user interface
2. A set of parameters editable via a set of sliders

Here's the parameter editor for the same plugin as shown above:



## Opening the Plugin and Parameter Editors

To open the plugin editor double click the plugin slot, or right click on it and choose "Plugin Editor":



To open the parameter editor, choose "Plugin Parameters" from the right click menu, or double click the plugin slot while holding the Alt key.

You can also toggle between the two editors by clicking the first button on the plugin editor toolbar.

## Other Plugin Editor Controls

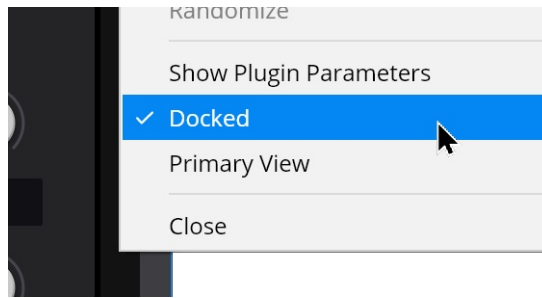
The plugin editor toolbar includes buttons and drop downs for the following:

- Toggling between the plugin editor and parameter editor
- Next and previous preset buttons
- Preset Selector drop down
- Lock preset button
- Solo plugin button
- Bypass plugin button
- Plugin gain control slider
- Menu with additional commands

## Normal, Detached and Docked Modes

The plugin editor can be displayed in one of several modes.

- Normally the plugin editor appears as a separate window from Cantabile's main window, always appears in front of Cantabile's main window and is minimized when Cantabile is minimized.
- By choosing "Detached" mode from the plugin editor menu, the editor appears independently of Cantabile's main window - either appearing in-front or behind it, and is minimized independently.
- By choosing "Docked" mode from the plugin editor menu, the editor is docked as a separate tab in the main work area of Cantabile's main window.



Docked mode is only available for the plugin's user-interface and not for parameter editing.

## Setting a Docked Plugin as the Primary View

When a plugin editor is docked it can be marked as the primary view. Whenever a song with a primary view set, it will automatically switch to that view when the song is loaded.

To set the primary view, first dock the plugin editor and then from the plugin editor menu, choose "Primary View".

Only one plugin can be marked as the primary view. Turning the primary view option off (by selecting the same command again) reverts the song to having no primary view.

## Morphing and Randomizing Plugin Settings

The plugin editor and parameter editor windows are also used for morphing and randomizing plugin settings. [See here](#) for more about this.

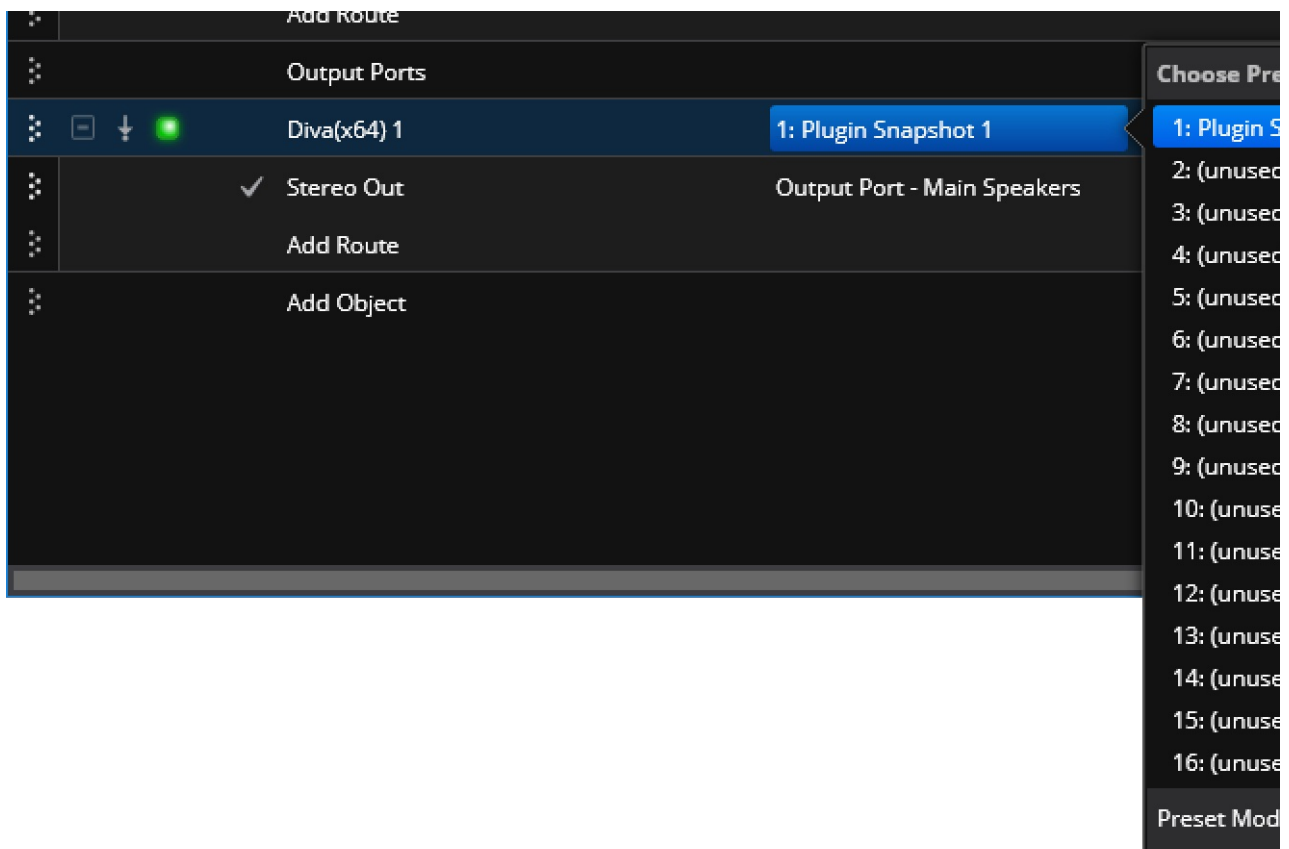
## Preset Models

Cantabile has three models for managing a plugin's presets. By selecting the appropriate preset model for a plugin you can balance preset switching speed and manage plugins that don't have a built in preset system.

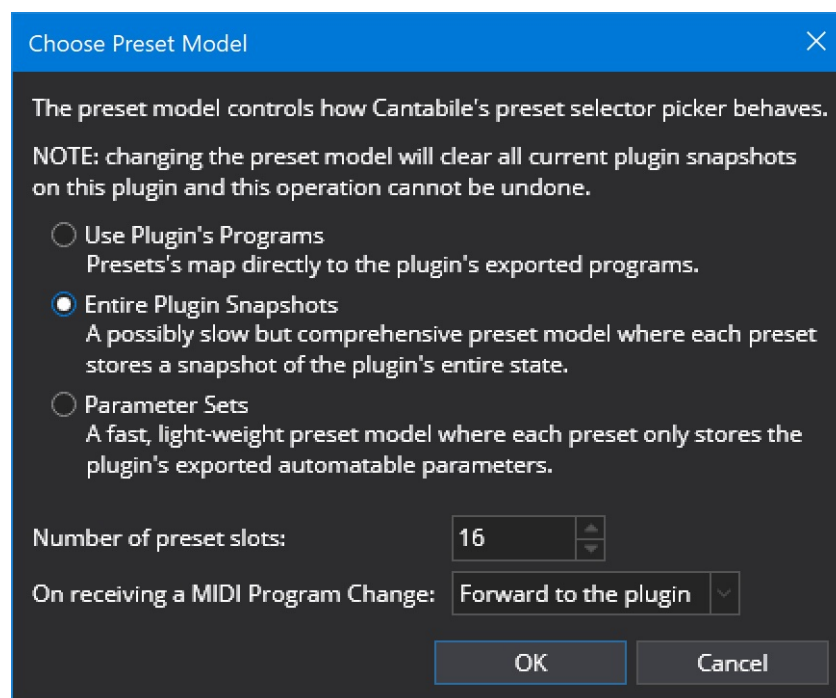
The selected preset model determines how a plugin's settings change when switching presets via Cantabile's preset selectors. This applies to the preset selector at the top of the plugin editor window, the preset selector in the plugin slot on the main window and the various [bindings](#) that work with presets.

When you open Canabile's preset picker the current preset model is shown at the bottom of the popup:





Clicking the "Change" button lets you change the preset model for that plugin instance:



Note that changing the preset model when the current model is "Entire Bank Snapshots" or "Parameter Sets" will clear the current preset bank and can't be undone except by reloading the song or rack.

## Choosing a Preset Model

The follow explains the different preset models and why you might choose to use one model over another.

### Use Plugin's Programs

Some plugins have a built-in set of presets (aka "programs") that they make available to the host. Selecting this preset



model leaves all preset management to the plugin itself.

This is the default preset model if the plugin indicates that it has more than 1 preset available for selection.

Note that some plugins have a comprehensive built-in preset system but don't make it accessible to the host program. For these plugins if you need Cantabile control over preset management you should choose one of the other preset models.

### **Entire Plugin Snapshots**

In this preset model Cantabile creates a bank of 128 simulated presets and each preset stores a snapshot of the entire plugin state.

This is the default preset model for plugins that indicate they only have one preset available.

If the plugin has its own internal preset bank, then each Cantabile preset will store a copy of the entire internal bank.

This preset model is the most flexible in that each preset stores an entire copy of the plugin's state however for slow loading plugins switching between presets can be slow. This preset model was formally referred to as "Pseudo Presets".

### **Parameter Sets**

This preset model creates a simulated bank of 128 presets where each preset only stores the values of the plugin's exported parameters.

Parameter Sets are a lightweight preset model. They are usually very fast to switch between (because they only tweak the plugin's parameters) however not all plugin settings are necessarily controllable through parameters.

This preset model is usually used with audio effects and modelled synths and may be less useful for sampled instruments.

## **MIDI Program Change Handling**

When a plugin receives a MIDI program change you can have the plugin respond in one of two ways:

- Select a preset matching the program change number
- Forward the program change to the plugin (handling will depend on the plugin)

Use the drop down at the bottom of the preset model window to change this behaviour.

## **Changing the Base Program**

When you've selected the "Entire Plugin Snapshots" or the "Parameter Sets" preset model, you might like to load one of the underlying presets exported by the plugin.

To support this, the plugin editor has an additional preset selector drop down button just to the right of the main preset selector.

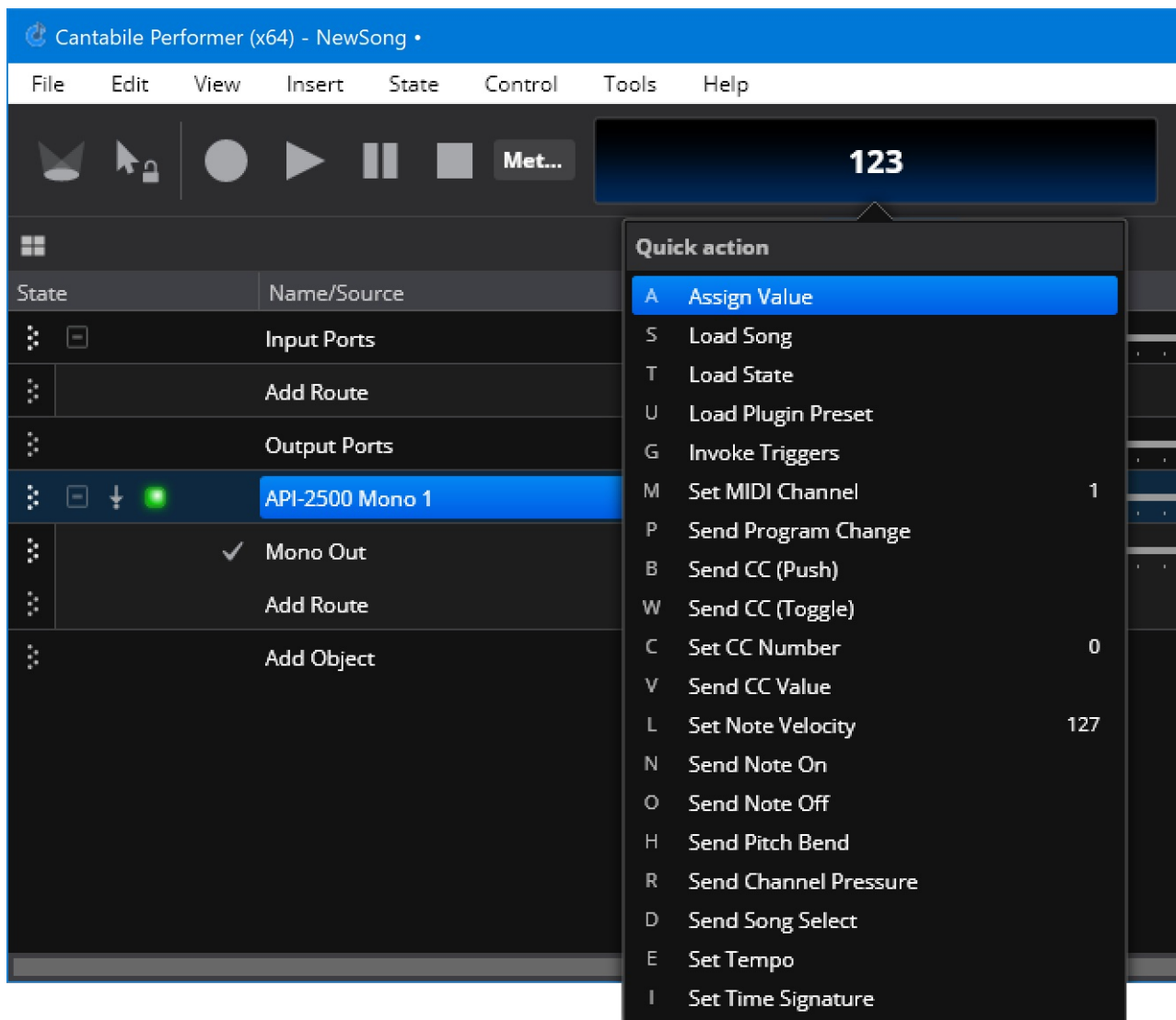


Selecting a preset from this drop down loads the selected preset into the current "Entire Bank Snapshot" or "Parameter Set" preset and optionally sets the preset name to match that of the plugin preset.

## Quick Controller

Cantabile's Quick Controller provides a fast way to load songs, states, presets and send simple MIDI events by entering a simple commands that consist of a number followed by a single letter command.

For example, to load the song with program number 123 you can type "123" and the letter 'S'.



## Using the Quick Controller

Using the Quick Controller is simple:

1. From anywhere on Cantabile's main window, start typing a number
2. Press the letter for the command to be invoked, or press Space, Enter or Tab to invoke some commands (see options below)

When you start typing a number, it appears in the status panel on the main toolbar and a menu appears showing the available commands and the associated letter.

## Options

There are a few simple options that determine how the Quick Controller behaves. These options can be found in the Options → Keyboard and Controls:

Quick Controller

Enter Key Action:

Assign Value

▼

Space Key Action:

Load Song

▼

Tab Key Action:

None

▼

☒ Show popup menu when active

The first three options determine which command is invoked when the Enter, Tab and Space keys are pressed. The checkbox supports disabling the popup menu (once you're familiar with the commands you need, you might find you don't need it).

Note that even with the popup menu disabled, you can still access it via the View | Quick Controller menu item.

## Available Commands

### Assign Command

Assigns the entered value to the selected numeric field. eg: If a song in the set list selected, its program number will be updated. Also works on states, plugin presets, bindings and trigger settings.

### Load Song

Load the song in the set list with a matching program number

### Load State

Loads the state with the matching program number

### Load Plugin Preset

Loads the entered plugin program to the currently selected plugin

### Invoke Triggers

Invokes all custom triggers with matching custom event number

### Set MIDI Channel

Sets the MIDI channel number of the on-screen keyboard and quick controller.

### Send Program Change

Sends a MIDI program change event. To send banked program change enter the bank number and program number separated by a period. eg: 1001.23P will select bank 1001, program 23.

### Send CC (Push)

Simulates pushing a MIDI CC button, sending value 127 followed by 0 to the entered CC number.

### Send CC (Switch)

Simulates switching MIDI CC switch, alternately sending 127 and 0 between each invocation.

### Set CC Number

Sets the CC number to be used for the Send CC Value command (below)

### Send CC Value

Send the entered CC value to the CC number selected by the Set CC Number command.

### Set Note Velocity

Sets the velocity to be used when using the Send Note On command

### Send Note On

Sends a MIDI note on event for the entered note number (using the velocity selected by Set Note Velocity command)

### Send Note Off

Sends a MIDI note off event for the entered note number

### Send Pitch Bend

Sends MIDI pitch bend event

### Send Channel Pressure

Sends a MIDI channel pressure event

### Set Tempo

Sets the metronome's tempo

### Set Time Signature

Sets the metronome's time signature. Separate each component with a slash. eg: entering "3/4I" will choose 3/4 time signature.

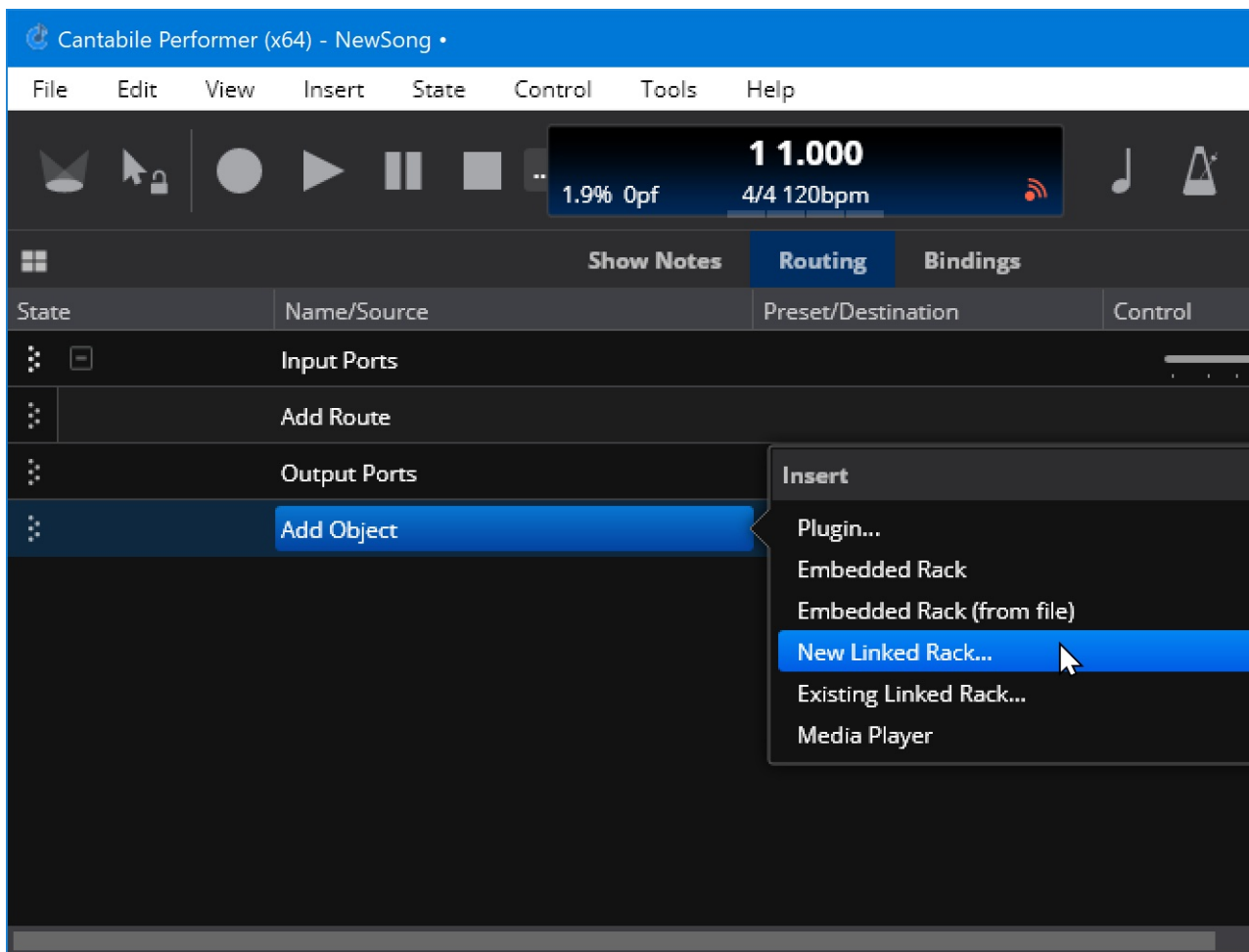
## Racks

Racks provide the ability to group together a set of objects into a self-contained "black box" that works as a standalone unit.

For example you could put an instrument plugin connected to a set of effect plugins into a rack. That rack would then appear in the parent song as a single self-contained slot wrapping all the plugins and their wiring.

## Inserting a Rack

To add a rack to a song, click the Add Object button and choose one of following commands:



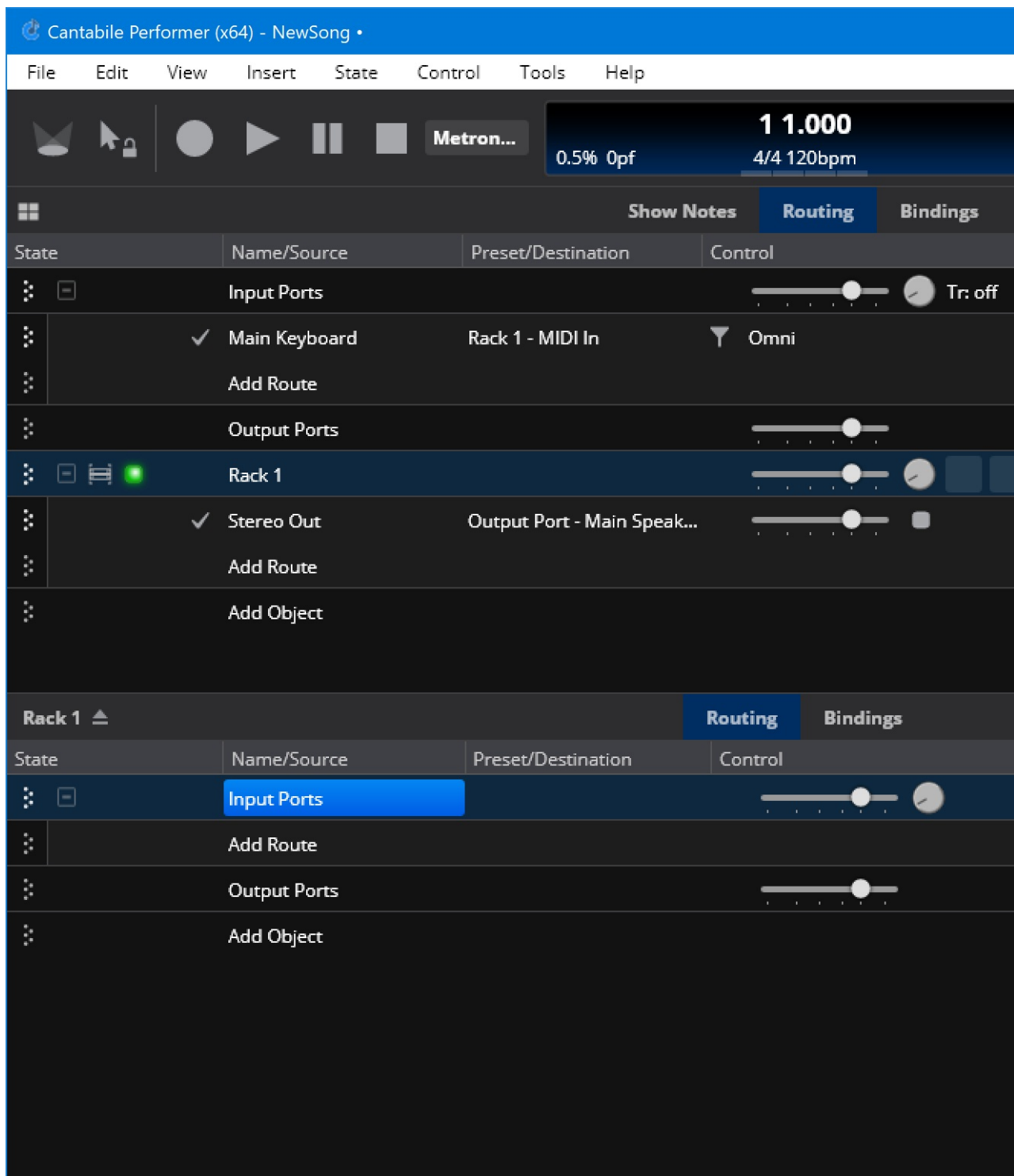
- Embedded Rack - inserts a new empty Embedded Rack
- Embedded Rack (from file) - inserts an Embedded Rack and initializes it with the contents of a saved rack file
- New Linked Rack - creates a new linked rack (will prompt for the name the rack should be saved as)
- Existing Linked Rack - select an existing rack file to be inserted as a linked rack

(See below for details on Linked vs Embedded racks)

## Editing a Rack

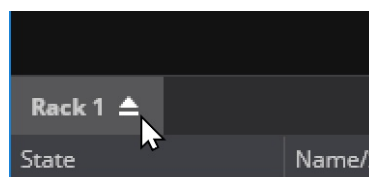
To edit the contents of a rack, double click its name in the parent song file (or select it and press Enter).

When editing a rack, Cantabile's main window splits and shows the rack's content in the lower half:



Editing the contents of a racks is identical to working with a song. Insert plugins, connect them using routes, setup bindings etc... exactly as you would for a song.

To close the rack click the "eject" button:



## Rack Ports

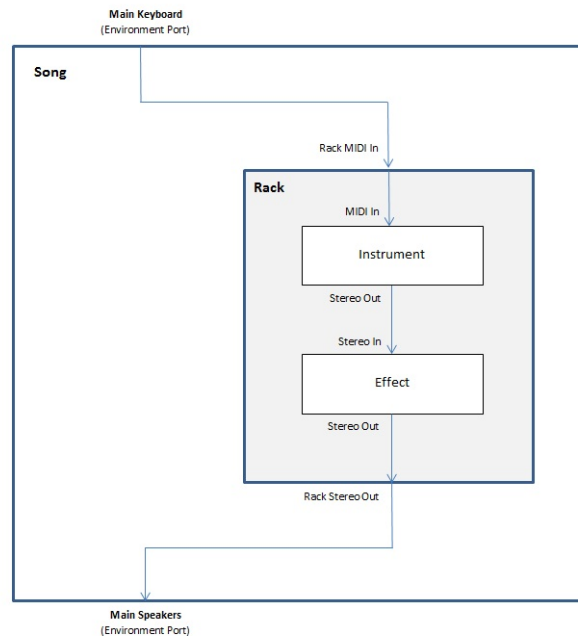
Racks support a set of audio and MIDI ports that can be used to create route connections between racks. By default, each

rack has the following ports:

- MIDI In
- MIDI Out
- Stereo In
- Stereo Out

Externally to the rack you connect your song and rack files by creating routes between the song and the racks ports. Internally you connect a racks ports to plugins and other objects in the rack.

For example, here's how a song with a single rack containing an instrument and an effect might be wired up.



Internally to the rack:

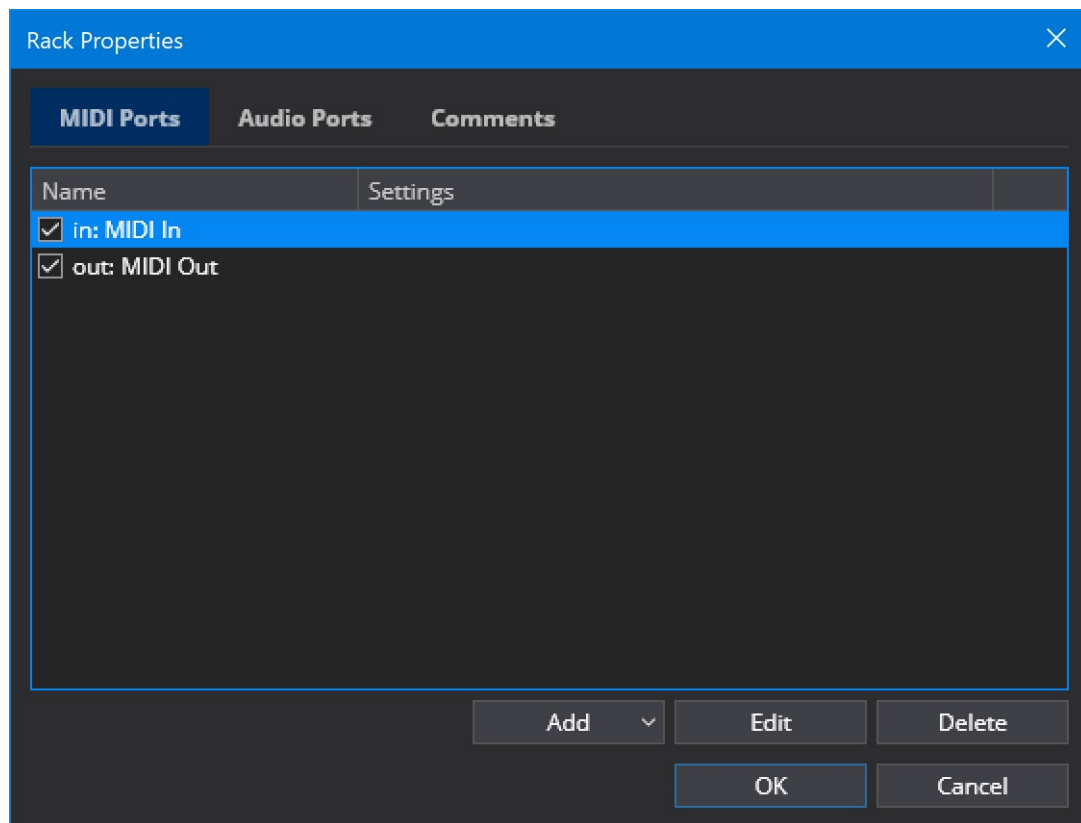
- The rack's MIDI In port is connected to the instrument
- The instrument is connected to the effect
- The effect is connected to the rack's Stereo Out port

Externally, in the song:

- The Main Keyboard environment port is connected to the rack's MIDI In
- The Rack's Stereo Out is connected to the Main Speakers environment port.

Although each rack has a default set of input and output ports, you can create your own port configurations by:

1. Opening the rack
2. From the File menu, select the Rack Properties command
3. Use the MIDI Ports and Audio Ports pages to configure the rack's ports



## The Background Rack

Aside from racks that you explicitly load into a song there is one special rack called the "Background Rack" that is typically used for bindings that need to work across all songs but can also be used to load plugins, media players or triggers that you need available in all sessions.

To access the background rack, from the View menu choose "Background Rack".

Changes to the background rack are always saved automatically when the Cantabile is closed.

## Linked vs Embedded Racks

Racks can either Embedded or Linked:

- Embedded Racks are saved as part of the parent song and can't be shared across songs (although they can be exported and re-imported into other songs)
- Linked Racks are saved in a separate .cantabileRack file and can be shared across multiple songs. They provide the ability to share plugin instances across multiple songs making song switching faster.

Linked racks are only supported in Cantabile Performer.

## Racks vs Songs

Racks and songs are very similar and working with a rack is almost identical to editing a song.

Songs and racks both support:

- Routing - a set of plugins, media players and nested embedded racks and the routes that connect them together.
- Bindings - assignments that automatically connect objects settings and MIDI devices.

Songs provide the same functionality as racks, but add:

- Tempo and Time signature settings
- A Global transpose setting - *Cantabile Performer Only*.
- Ability to load linked racks

Note that only songs can load linked racks. Racks can have embedded racks but can't load linked racks.

A song is the primary file you work with at any one time in Cantabile and only one can be loaded at a time.

## Song and Linked Rack Lifetimes



The "lifetime" of songs and racks is handled differently.

When switching songs the song and all routes between the song and it's loaded racks are immediately closed and any held notes released. Embedded racks are closed when the parent song is closed.

Linked racks only closed when the current song no longer refers to them. If you switch between two songs that use the same rack, that rack will be kept loaded and continue to process uninterrupted.

## Song and Rack Pre-loading

You can instruct Cantabile to pre-load all songs and racks in a set list in order to improve song switching times. See [Set Lists](#).

## Song and Rack States (Cantabile Performer Only)

In Cantabile Performer, both songs and racks support [States](#) however they're typically used for different purposes:

- In racks, states are typically used to switch between different sounds where each state works similarly to a plugin preset and is selectable from the parent song's just like a plugin preset
- For songs, states are typically used to represent the parts of a song (eg: intro, chorus, verse).

The terms "Song State" and "Song Part" are used interchangeably in Cantabile.

## Per-Song Rack Settings (Cantabile Performer Only)

When editing the state behavior of objects in a rack you can indicate that the setting should be stored on a per-song basis. This allows for control over a rack's settings on a per-song basis without having to create states for every possible combination. See [States](#) for more information.

# Recording

*Cantabile Solo and Cantabile Performer Only.*

Cantabile supports audio and MIDI recording, including automatic recording where the recorder starts and stops automatically as you play. Cantabile can be left in auto record mode and it will capture everything you play.

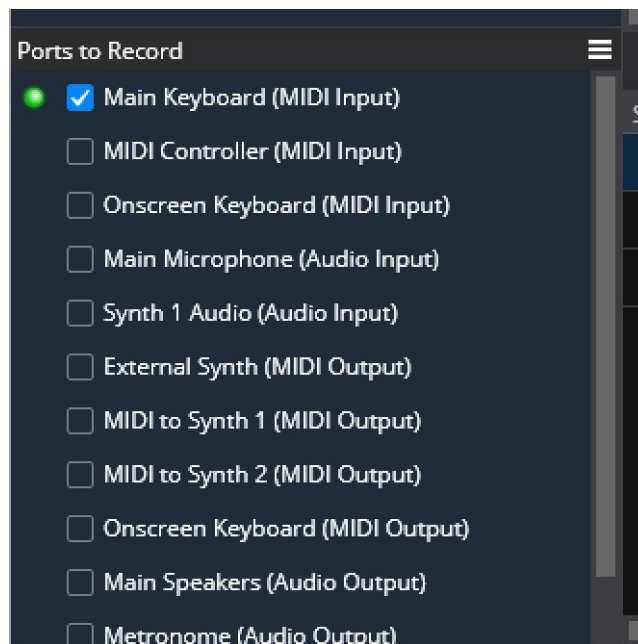
This [video walkthrough](#) also demonstrates working with the recorder.

## Selecting what to Record

The recorder captures audio and MIDI data from a set of input and/or output ports that you select. For example you might choose to record incoming MIDI from your main keyboard as well as audio sent to the speakers.

To select the ports that should be recorded, from the View menu, choose "Recorder Ports". This will show a small panel at the bottom of the recordings list where you can choose which ports to record.

- Place a checkmark next to any ports that should be recorded
- Select the green LED indicator to arm the port for auto-recording



Note that a port doesn't need to be connected to a physical device to record it. This can be handy to record signals from inside the song - create an audio or MIDI output port, leave it disconnected, setup a route to send the signal to be recorded and then select the port in the recording ports list.

## Manual Recording

To start recording, click the Record button on the main toolbar, use the shortcut key ('R' by default) or choose the "Record" command from the "Control" menu.

Recording will continue until you stop the recording by pressing the Record button again. Once the recording has completed, a new entry will appear in the recording list.

Recordings with a duration of less than 5 seconds will be discarded. You can adjust this duration in Recording Options.

## Automatic Recording

To enable automatic recording, from the Control menu choose "Auto Record", or right click on the Record button on the main toolbar and choose "Auto Record".

When auto-record is enabled, a green ring around the record button will light up.



For a port to be monitored for activity, it must be both selected for recording and armed for auto recording.

Auto record will stop recording after 5 second of silence. When monitoring audio ports, there is a noise threshold setting that can be used if ambient noise is triggering the audio recorder. Both of these settings can be adjusted in Recording Options.

## Recording Files

Each recording will be saved as one MIDI file and one Audio file (unless no ports of a particular type are enabled).

For MIDI files each port selected for recording will be written to a separate MIDI track. The track will be named using the port name.

For Audio files each port will be written as a consecutive set of channels. For example, if you have two stereo ports selected the resulting wave file will have four channels.

## Pinned and Unpinned Recordings

In the list of recordings, each recording has a "pin". Unpinned recordings will be deleted after 7 days (adjustable in Options).

Pinned recordings are never automatically deleted.

By default all recordings are pinned and will be kept indefinitely. In Recording Options, you can choose to not pin new recordings. In this case all recordings will eventually be automatically deleted - unless you manually pin them.

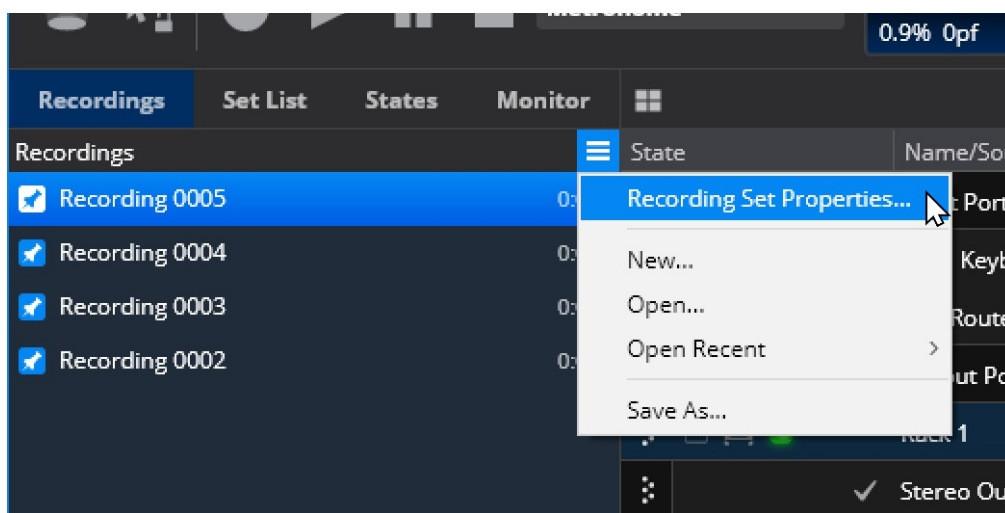
## Recording Sets

The list of recordings shown in the recordings tab is saved as a "Recording Set".

A recording set includes:

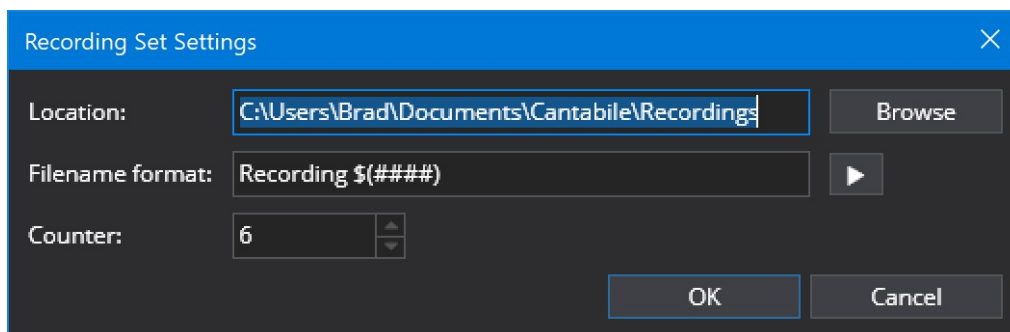
- The list of previous recordings
- The set of ports to record
- The set of ports armed for auto-record
- A set of properties that control where to save the recordings, filename formats etc...

To adjust the recording set properties, from the recording panel menu choose "Recording Set Properties":



You can also use this menu to create new recording sets or load previous recording sets.

Recording sets have the following properties:



## Per-Song Recording Sets

Usually Cantabile will keep the same recording set open regardless of the currently loaded song file.

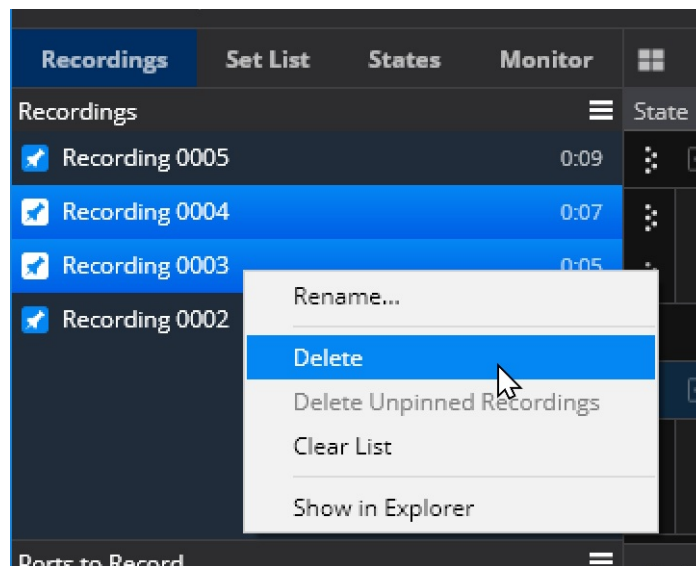
If you'd like to create separate recording sets for each song, go to Tools → Options -> Recording and enable the option "Automatically create a new recording set for each song".

When enabled each time you create a new song, or open an existing song and start a recording a recording set will be automatically created for that song.

## Working with Recordings

The recordings list shows a list of all previous recordings. You can use this list manage your recordings. The right click context menu includes commands to rename, delete, re-order and launch Windows Explorer with the recording selected.

There are also commands to delete all unpinned recordings as well as to clear the list without deleting recordings.



## Recording Filenames and Locations

You can change where recordings are saved and the name of recording files in Options.

Options

General

Audio Engine

Audio Ports

MIDI Ports

Plugin Options

Recording

Startup/Shutdown

File Locations

Keyboard & Controls

Colors

Hot Keys

External Tools

Miscellaneous

Diagnostics

About

Recording Sets

☐ Automatically create a new recording set for each song

Recording Set Defaults

Location:

C:\Users\Brad\Documents\Cantabile\Recordings

Filename format:

Recording \$(####)

These settings will be used as the defaults for new recording sets.

To set the default port configuration, use the "Set as Default" command in the recorder ports panel menu.

File Format

MIDI Time Format:

Realtime (Compatible)

Audio Sample Format:

16-bit integer

Wave File Type:

Automatic

Auto-record

Stop auto-recordings after

5.0

seconds silence

Discard recordings less than

5.0

seconds long

Audio noise threshold:

5.00

%

Pinned Recordings

☒ Pin new recordings

Remove pinned recordings from list after

7

days (set to zero to disable)

Delete unpinned recordings after

7

days (set to zero to disable)

OK

Note: the "Recording Set Defaults" section sets the defaults for how new recording sets are created - it doesn't affect the currently loaded recording set.

## MIDI Time Format Options

When recording MIDI, several different time formats are available:

### Realtime (Compatible)

This is the default recording format that is most compatible with other software. It records in real-time (ie: each event is timestamped with millisecond accuracy). Tempo and time-signature information isn't captured in this mode.

### Realtime (SMPTE)

This format is very similar to Realtime (Compatible) except the file is marked as SMPTE time format. This is a more correct format for this type of recording but some software programs don't support it.

## Musical (Relative) and Musical (Absolute)

In these formats events are timestamped using musical time format.

This format is best for importing into other software programs where maintaining correct musical time is more important than millisecond accuracy. In this format the tempo and time signature of Cantabile's master transport is also recorded (even if it changes during the recording).

In "relative" mode events will be timestamped such that the recording starts just before the first recorded events. In "absolute" mode events will be timestamped using the exact time of the master transport.

To better understand the difference between relative and absolute modes, consider this scenario: suppose a recording is started when the current master transport is at bar 50 (perhaps because the metronome has been running for a while).

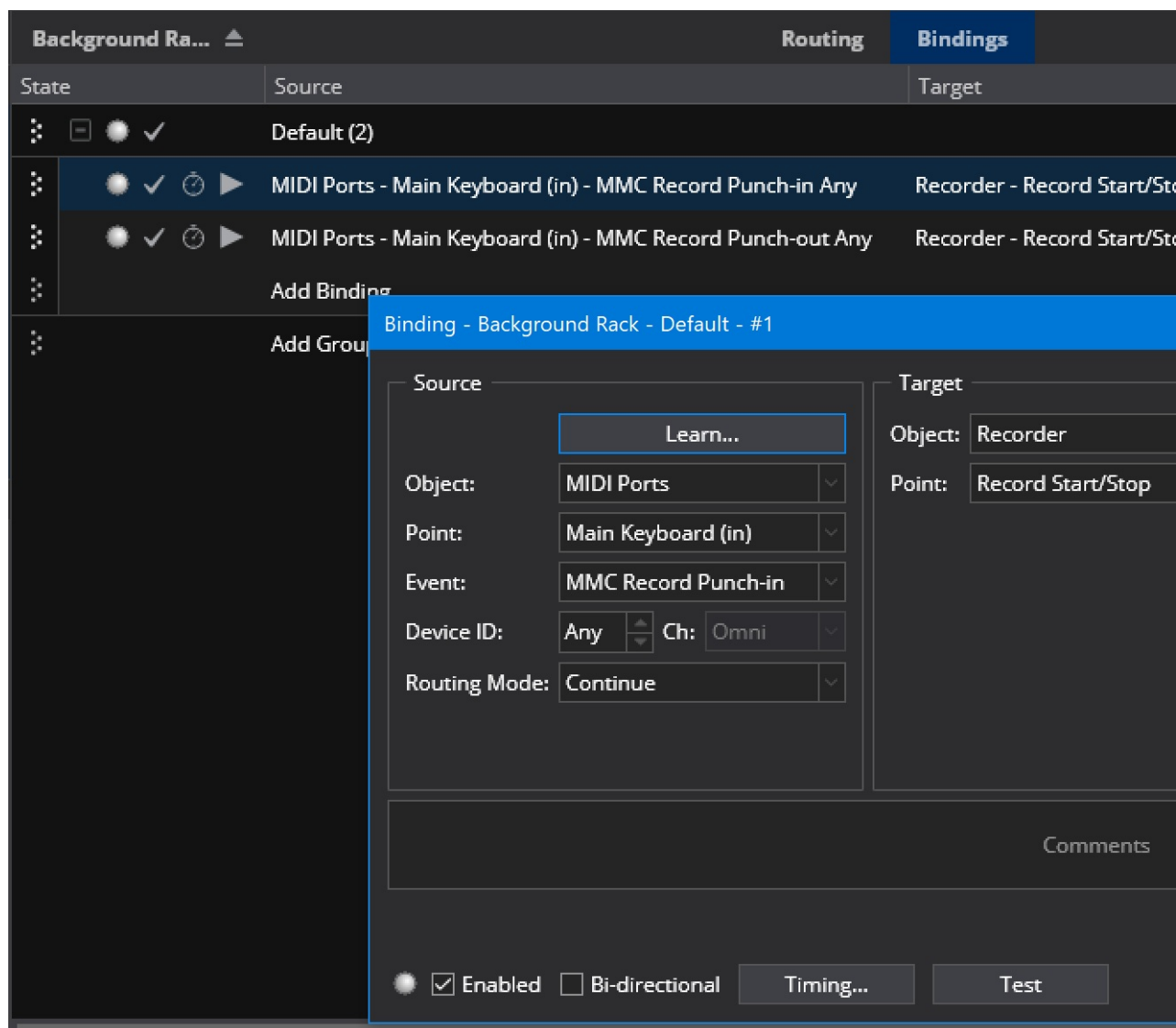
- In absolute mode, the recording would have 50 bars of silence at the start.
- In relative mode, the recorded events would be at the start of the recording.

Note that the musical recording formats require a running master transport. If the master transport isn't playing when the recording starts, the recorder will automatically start it (if it can). Similarly, if the master transport stops while a recording is in progress, the recording will also stop.

## Bindings

The recorder can be externally controlled using Bindings.

ie: you can assign buttons on your keyboard to start/stop recording, enable/disable automatic recording, pin/unpin the last recording etc...

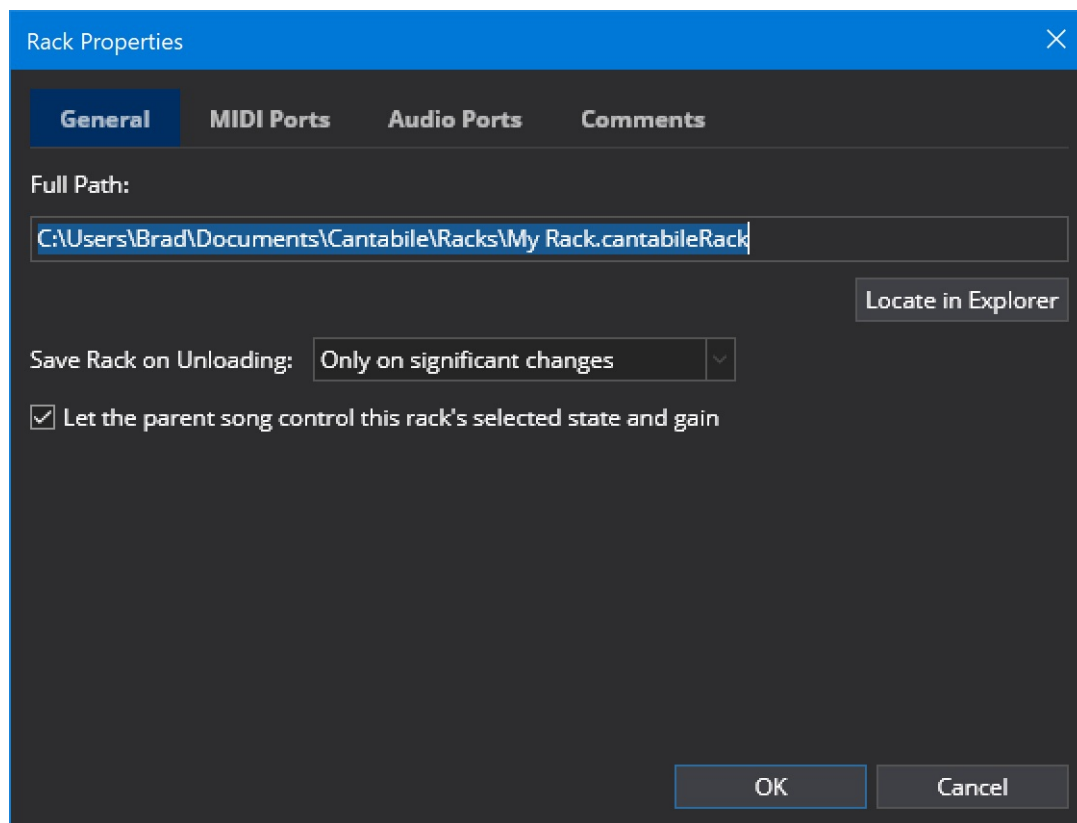


See [Bindings](#) for more.

## Song and Rack Properties

Every song and rack file in Cantabile has a few options that control how it behaves.

To adjust these settings, open the song or rack file and from the File menu choose "Song Properties" or "Rack Properties" (depending which type of file is currently active).



### Full Path

Displays the full location path of the saved file

### Locate in Explorer

Opens Windows File Explorer and selects the song or rack file

### Prompt to Save

Controls the kinds of changes in the file that cause it to be marked as modified (See below)

### Always Save When Saving the Parent Song (Rack Options Only)

Forces this rack file to be saved when the parent song is saved - even if it hasn't been marked as modified.

### Let The Parent Song Control This Racks Selected State and Gain (Rack Options Only)

Normally on, this option lets the parent song of rack control it's state and gain settings. Turn this option off for racks that are intended as a global output control. By turning this option off settings in the rack are isolated from the parent song and can be used to create global changes across all songs. eg: you could put an EQ in such a rack, and route all outputs through it and then tune the EQ for a particular venue.

### Comments (Separate Tab)

A set of comments about this song or rack. What it is, how to use it etc...

### Automatically Show Comments When Rack is Inserted (On Comments Tab)

When enabled, the comments will be shown to the user when the rack is inserted into a song.

## Prompt To Save Options

The Prompt To Save option mentioned above has follow choices:

### Never

The file will never be marked modified and therefore you'll never be prompted to save it

### Only On Significant Changes

Only changes to the structure of the file cause the file to be marked modified. eg: inserting a plugin would mark it as modified whereas tweaking a plugin parameter wouldn't. This option is handy for working with plugins that generate excessive change notifications causing the file to always be marked as modified.

### On Any Change Except via Bindings

Any change will mark the file as modified - unless the change occurs as the result of a binding.

### On Any Change

Any change will mark the file as modified.

See also [Preventing Prompts to Save](#).

## Stream Deck Integration

*Cantabile Performer Only*

The Cantabile Stream Deck plugin provides tight integration between Cantabile and one or more Stream Deck controllers.

Note: this is a plugin that's loaded into the Stream Deck App, not Cantabile.







## Features

Included actions:

- Audio Engine Power On/Off
- Live Mode
- All Sounds Off button (panic)
- Scrollable list/page of songs (names shown on buttons)
- Scrollable list/page of states (names shown on buttons)
- Countdown timer with presets and remaining time shown on button
- Master Transport controls (with icons that light up to show current state)
- Record and Auto Record controls
- Tap Tempo button (with tempo flasher)
- Metronome controls (tempo, time signature, sound on/off)
- Media Player by Index controls (play, pause, stop, play/pause, play stop, fast-forward/rewind, next/previous marker)
- Gain controls (Master in/out, audio ports, metronome)
- Send MIDI from a stream deck button
- Tightly integrated with the onscreen keyboard controllers (reflect controller state on the stream deck for notification/cue lights etc...)
- Show Notes controls (page up/down, line up/down, home/end etc...)
- Transpose controls (up/down semitone, up/down octave, up/down octave snapped)
- String Expressions (display Cantabile \$(variables) on a button)
- Invoke any UI command
- Invoke any binding target

## Compatibility

The plugin has been tested with the Stream Deck standard (15-button) and the Stream Deck XL (32-button). It mostly works in the Stream Deck mobile app but I don't recommend it - the choose state and choose song actions don't work (not sure why yet) and it's a bit laggy (especially for the tempo flasher). It ok for getting a feel for what the plugin does but that's about it.

## Installing the Plugin

As yet, the Cantabile Stream Deck plugin isn't available from the Elgato Stream Deck store and has to be manually installed.

To install the Cantabile Stream Deck plugin.

1. Make sure you have Cantabile build 4046 or later installed ([download](#)). Although the plugin will work in a reduced capacity on older versions (even v3), it's highly recommended to use the newer build to ensure all features work correctly and reliably
2. Enable Cantabile's network server - [see here](#) for more on this
3. Download the Cantabile Stream Deck plugin [from here](#)
4. Double click the downloaded file on the same machine that has the Stream Deck app installed and it should install itself

## Upgrading the Plugin

In order to upgrade an existing installation of the Cantabile Stream Deck plugin you must first manually remove the prior installation as follows:

1. Quit the StreamDeck app
2. In Windows File Explorer, in the address bar, go to the folder %APPDATA%\Elgato\StreamDeck\Plugins
3. Delete the folder com.toptensoftware.cantabile.sdPlugin
4. Repeat the instructions for Installing the Plugin above
5. When prompted to install profiles choose No (otherwise you'll get duplicates).

## Known Issues

- The Choose State and Choose Set List actions don't work correctly on the Stream Deck mobile app
- Tempo flashing in the mobile app is too laggy to be useful

## Included Profiles

The plugin includes several Stream Deck profiles for both the standard 15 button Stream Deck and the 32 button Stream Deck XL:

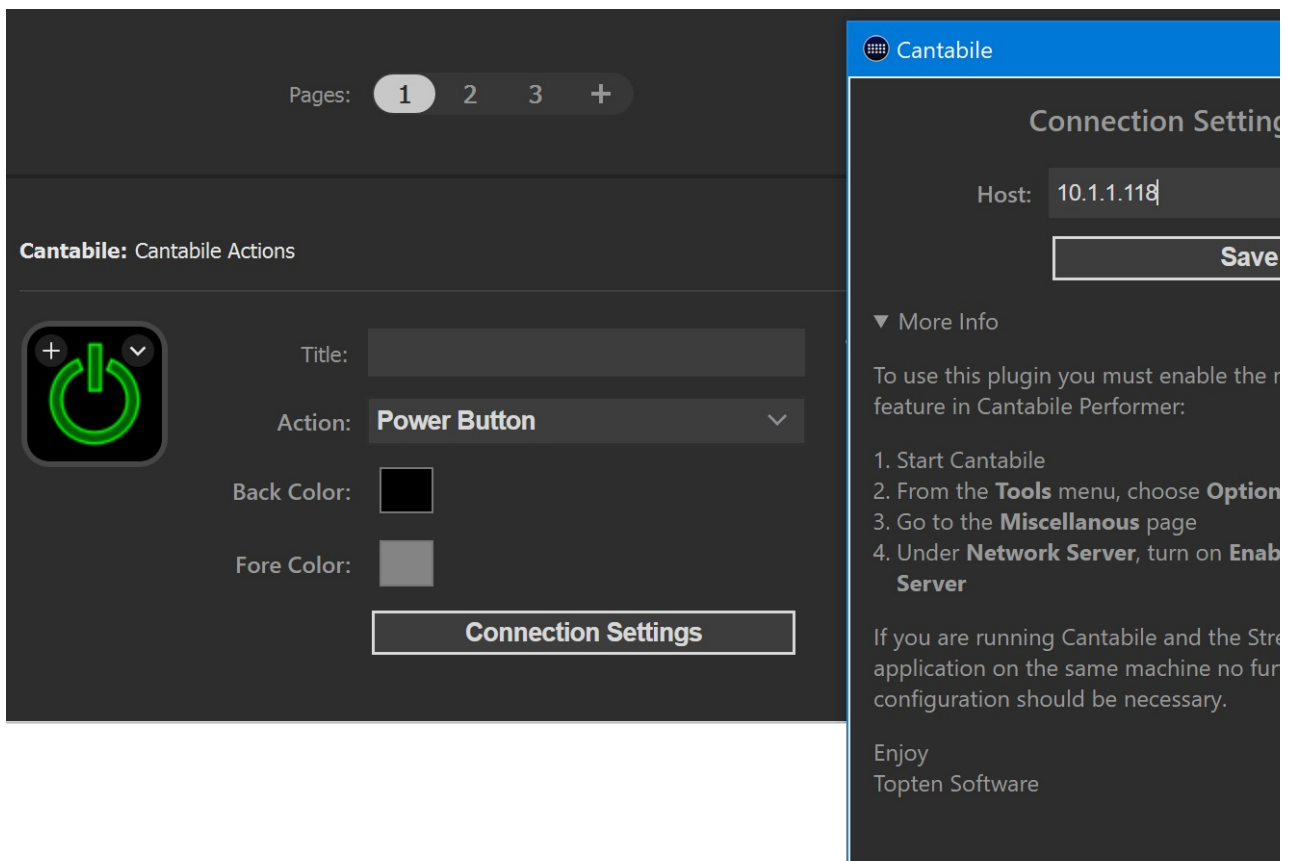
- Cantabile - a profile with common actions to serve as a starting point for configuring your own setup.
- Cantabile Choose Song - a read-only profile used by the Set List | Choose Song action
- Cantabile Choose State - a read-only profile used by the States | Choose State action

Profiles for the Stream Deck mobile app and foot controller are not included at this stage.

## Configuring the Connection

By default the plugin connects to a Cantabile server running on the same machine as the plugin (aka: 'localhost'). If Cantabile is running on a different machine:

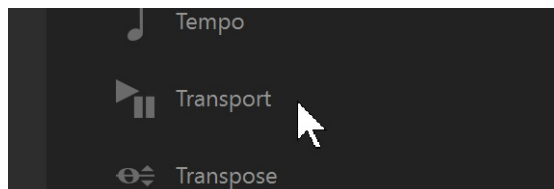
1. In the Stream Deck app, click on the "Power Button" action
2. In the properties for this action there's a button "Connection Settings"
3. Click the button to enter a new host address
4. Click Save to save the setting and connect to the new machine



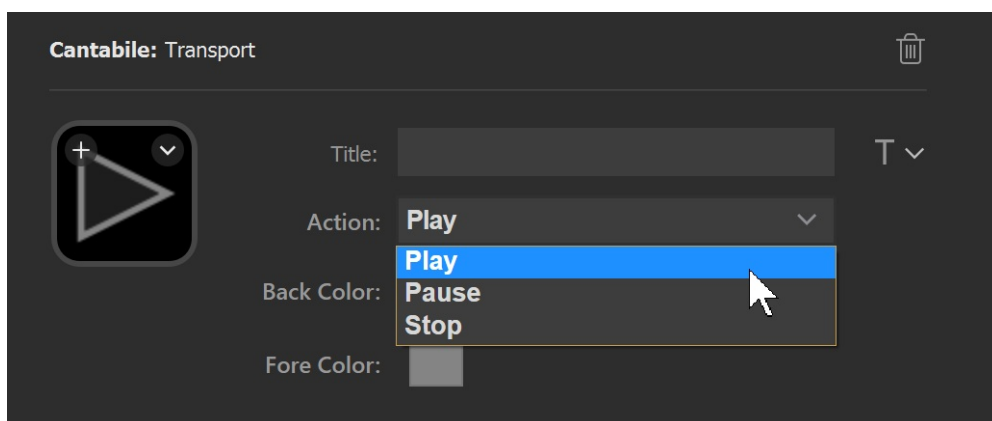
## Organization of Actions

The Cantabile Stream Deck items are grouped according to function with an action setting to control what the button actually does.

For example, the Transport item:



Has sub-actions for Play, Pause and Stop:



## Button Appearance

The Stream Deck application allows you to change the image and title on a button. Please be aware that this will overwrite any images and titles that the plugin is trying to show. This means that any button that provides feedback from Cantabile

may not work correctly.

For example:

- If you set an image on the tempo button, it will not flash
- If you set a title on the tempo button, it won't show the current time signature and tempo
- If you set an image on the play or pause buttons, they will still work but won't light up to show the playing or paused state
- If you set an image on a load song or state button it won't show the song or state name
- etc...

In short, it's not recommended to set custom images or titles on buttons. For this reason, all of the plugin actions have properties to set the foreground and background colors.

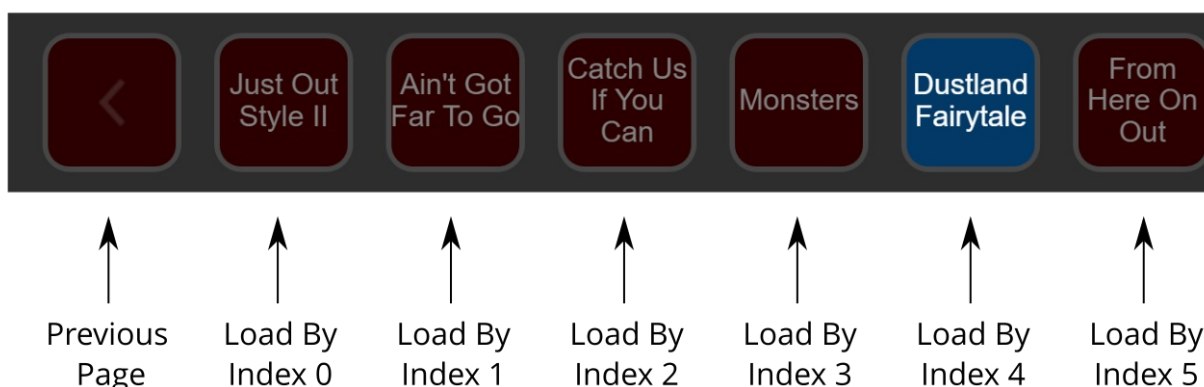
## Page Controls

The Set List and State actions both provide the ability to build a scrollable list of items (either songs or states) using the "Load By Index", "Next Page" and "Previous Page" sub-actions.

Setup this feature as follows:

1. Add a button with the "Load By Index" sub-action and set its index property to 0. This will represent the first item on the current page.
2. Repeat step 1, with increasing values for the index (0, 1, 2, 3 etc...) for each button on the page.
3. Add a button with a sub-action "Next Page" and set the page size property to the number of "Load By Index" buttons you created.  
ie: if you created 6 load by index buttons (with indices 0, 1, 2, 3, 4 and 5) the page size should be set to 6.
4. Add a button with the sub-action "Previous Page".

Once configured correctly, the Load By Index and Next/Previous page buttons work together to create a scrollable list of items.



Note that most of the properties of the Load By Index sub-action are linked so changing the setting on one button will also affect all the other similar buttons on the same screen. This is by design and simplifies configuring these screens.

## The MIDI Action

The MIDI button action can be used to send MIDI to any of the output MIDI ports configured in Cantabile.

It can also be configured to inject MIDI into Cantabile's on-screen keyboard device such that it appears the MIDI is originating from the on-screen keyboard device (and any ports attached to it). Because this device is owned and managed by Cantabile it provides for tighter integration with the Stream Deck plugin:

When "Device: Onscreen Keyboard" is selected as the MIDI target:

- Toggle Buttons mapped to controllers (including CC, FineCC, RPN, NRPN etc...) will always reflect the active state of the controller
- Buttons configured to send program changes (normal or banked) will light up when that program is selected for the on-screen keyboard

In both cases, it doesn't matter if the underlying setting is changed from the StreamDeck, from Cantabile's controller bar,

from bindings or from other network clients... the Stream Deck button will always reflect the current setting.

## Tips

Most of the other button actions are self explanatory, but here's a couple of tips:

- Long pressing (about 1 second) the Record button toggles auto record on/off.
- The tempo button works as a Tap Tempo button (tap it at the rate of the desired tempo).
- When the countdown timer is running it can be reset by a long press.
- When the countdown timer is stopped, a long press will cycle through the time out periods configured for the button.
- The power button has a Connection Settings popup that can be used to configure the connection to Cantabile.

## Debugging

In order to assist with diagnosing issues you may be asked to check the debug console for the plugin. This is a little cumbersome to setup, but can provide invaluable diagnostic information.

First, you need to enable debugging of the plugin:

1. Start the Windows registry editor by clicking the Start button and typing "regedit"
2. Navigate to the key: HKEY\_CURRENT\_USER\Software\Elgato Systems GmbH\StreamDeck
3. Right click in the right-hand panel and choose New → DWORD (32-bit) Value
4. Name the key `html_remote_debugging_enabled`
5. Set the key value to 1.
6. Restart the Stream Deck application (right click tray icon and choose "Quit Stream Deck")

	hasBeenLaunchedBefore	REG_SZ	true
	html_remote_debugging_enabled	REG_DWORD	0x00000001 (1)
	InstallDir	REG_SZ	C:\Program Files\Elgato\S

Once debugging has been enabled:

1. Reproduce the problem
2. Bring up any Chromium based browser (Chrome, Edge, Opera)
3. Navigate to <http://localhost:23654>
4. In the list of inspectable pages, click on the item labeled "Cantabile Stream Deck Plugin"
5. Click on the Console tab at the top to view the log
6. Right click on the log and choose "Save As" to save the log file.

Once finished you can set the registry key back to 0.

## Song and Rack States

*Cantabile Performer Only*

States provide a way to save the state of a song or rack and then recall those states to quickly switch between different configurations. States can be used to control most route settings, plugin presets and parameters, metronome settings, gain and mix levels and more.

The main purpose of states is to provide a fast mechanism for switching between the required sounds and settings for different songs or song parts.

This [video walkthrough](#) also demonstrates how to use states.

## Terminology

The term "state" refers to the general ability to take a snapshot of a song or rack and save it with a supplied name.

Although songs and rack both support states, the typical use case for each is different:

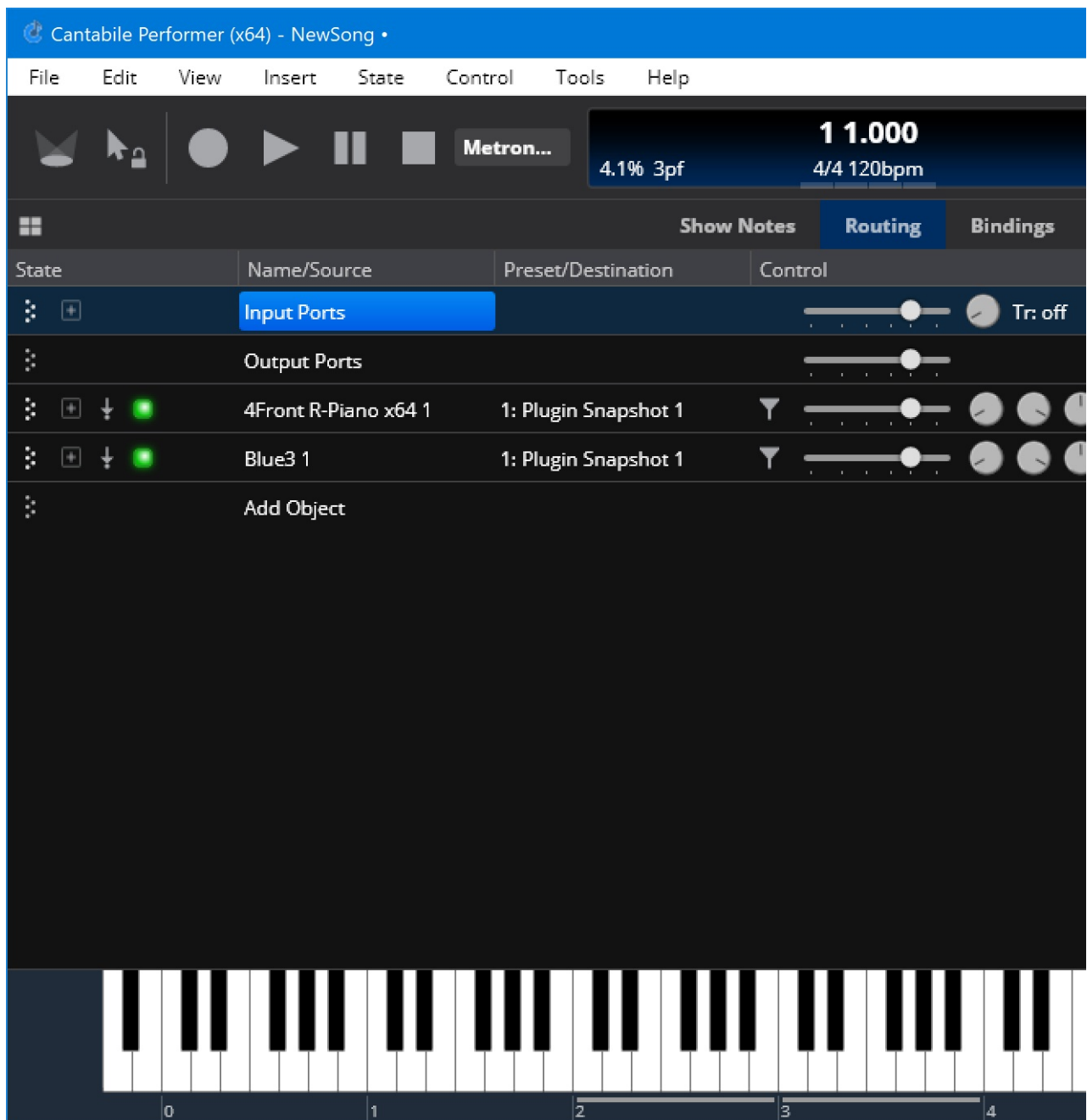
- For racks, states are typically used like plugin presets where each state represents a different sound.
- For songs, states are typically used to represent the different parts of the song - intro, chorus, verse etc...

For this reason song states are often referred to as "Song Parts". Song Parts and Rack States are functionally identical.

If you're familiar with Cantabile 2 you should also note that "States" is the new name for what used to be called "Sub-Sessions".

## Walk-through

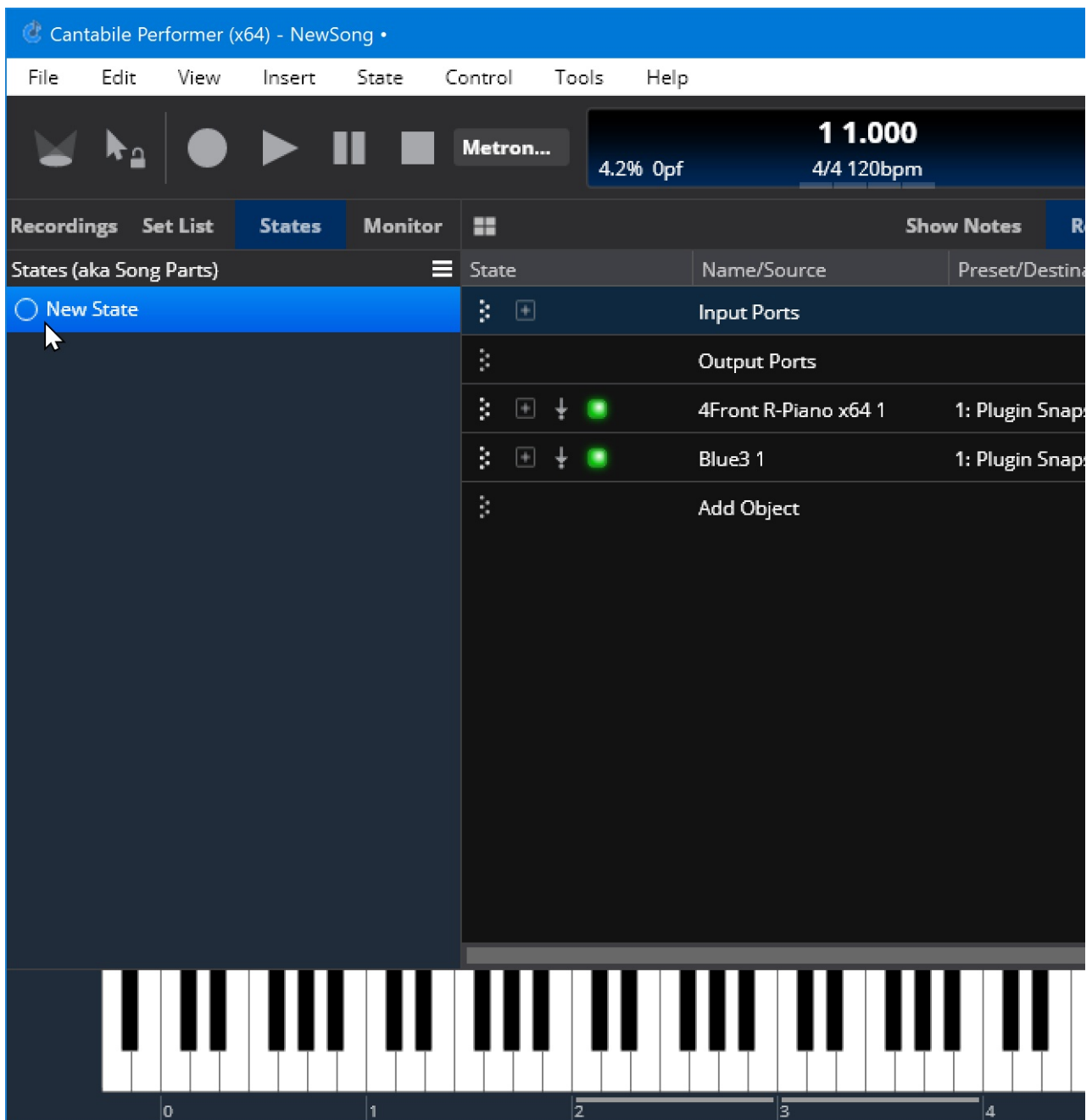
The easiest way to understand states is with a simple walk through. First let's setup a simple song with two instruments:



The rest of this guide will setup the states so that the one keyboard can be used play both instruments by quickly switching between them.

## States Panel

States are managed in the States panel which can be activated using the View|States menu command or by pressing Ctrl+T:



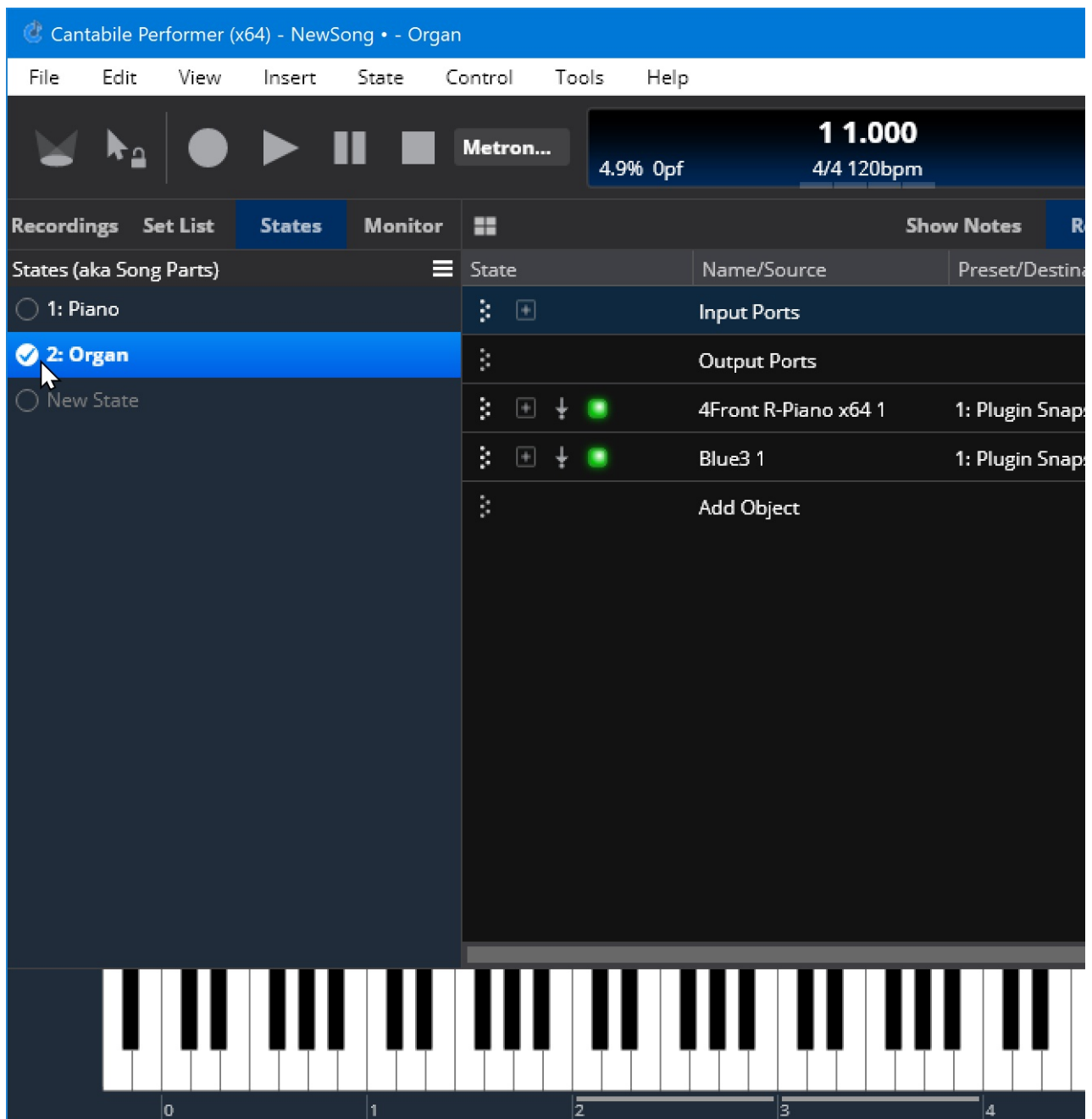
You can hide the state list by resizing it down until it disappears, or by activating it and pressing Shift+Escape.

*(Note that although we're editing the song's states, the panel is labelled "Song Parts" to suggest how states on a song are intended to be used)*

## Creating and Configuring States

We will now configure two states to switch between the two instruments.

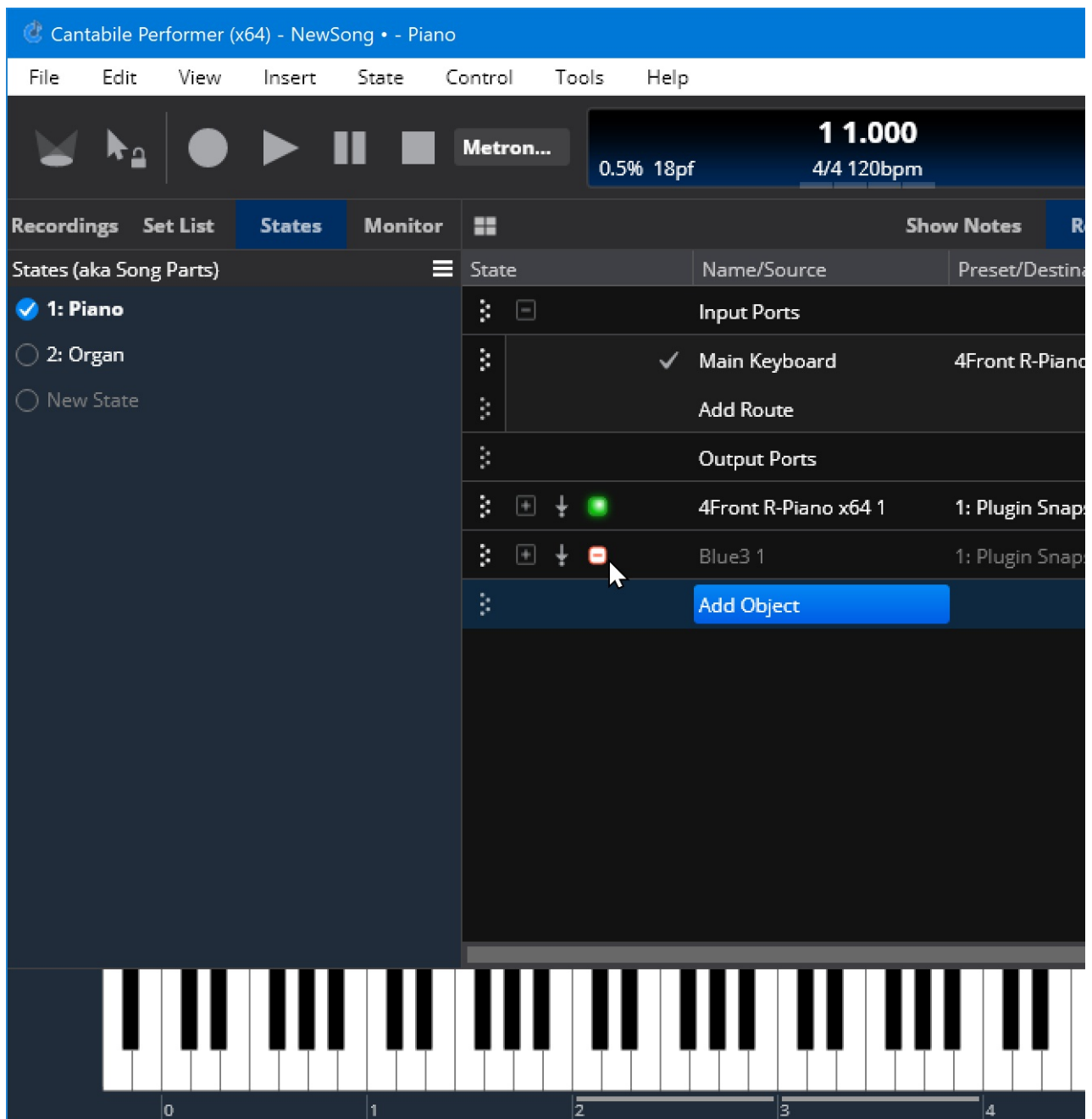
To create a state select Insert|State from the main menu, or click the little circle next to "New State" in the state panel list. You'll be prompted to enter a name for the state after which the current state will be captured and the state will be added to the state panel list. Do this twice to create two states:



Now switch back to the first state "Piano" by clicking the circle next to it. Configure the state by:

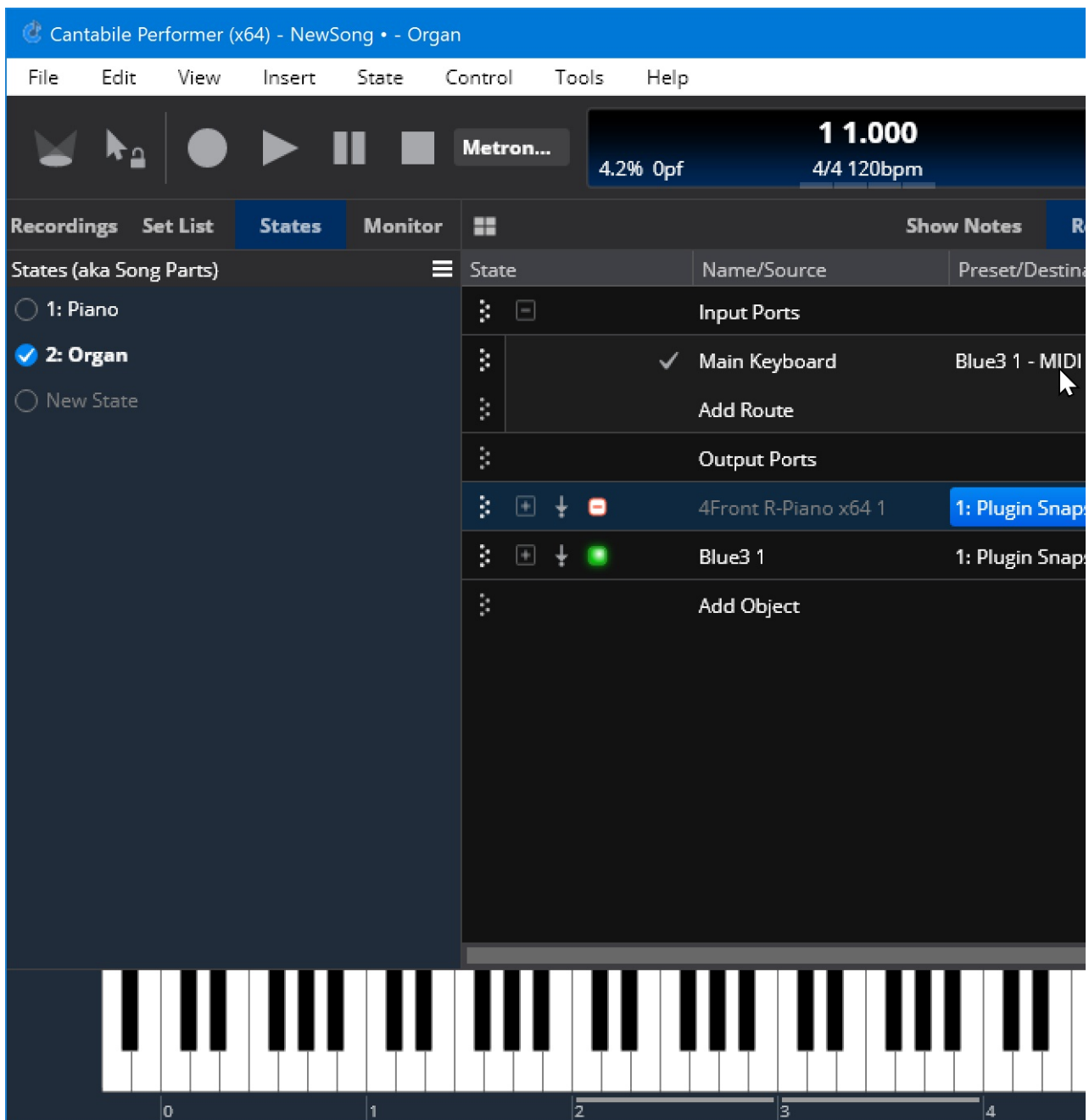
1. Delete the second input MIDI route - we'll be using the one route for both instruments.
2. Switch the synth plugin to suspended mode since we won't be using it in the "Piano" state. (click the green active indicator to make it orange).





Repeat the same process (but opposite) to configure the "Organ" state:

1. Change the MIDI route's target to the organ plugin instead of the piano.
2. Suspend the piano plugin and resume the organ.



## Switching Between States

Once you're states are setup, there are different ways to switch between them:

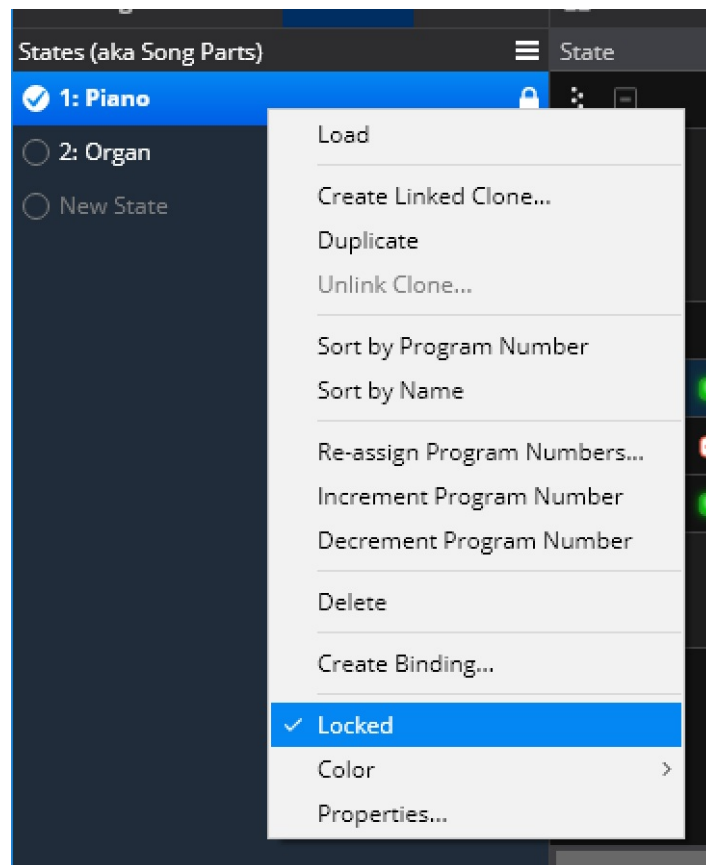
1. click the radio button (circle) to the left of the state name
2. double click the list entry
3. select the list entry and press Enter to load it
4. use the Next State/Previous State commands from the Control menu
5. press the T and Shift+T keys to move to the next or previous state
6. using [Bindings](#)

If you switch between the two states and play notes from an attached keyboard you should hear a different sound on each state.

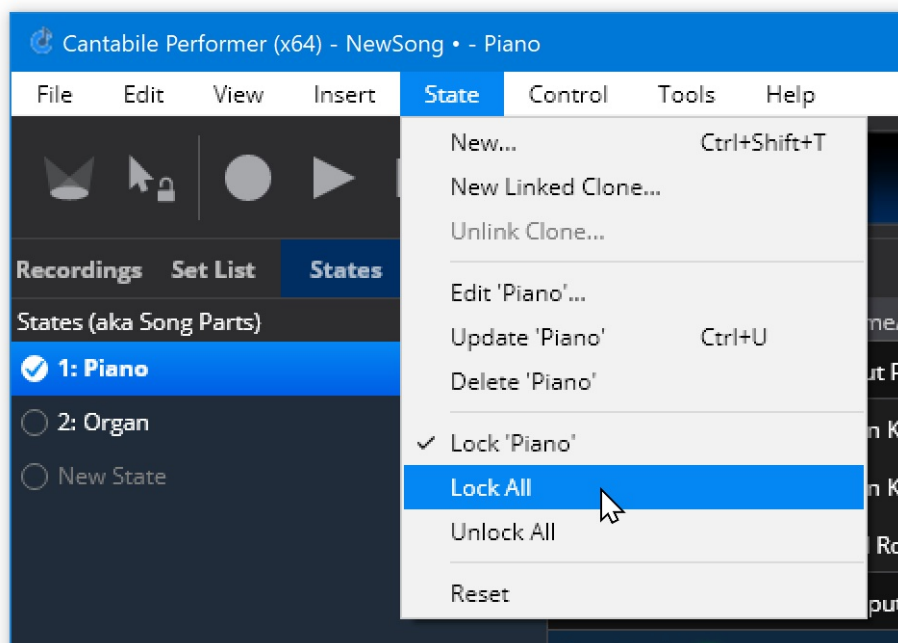
## Locked States

Normally Cantabile will automatically re-save the current state when switching to another state. This makes it easy to setup your states by just switching between them and configuring things as you need them.

Once you've got a state setup correctly however you might like more explicit control over when the state is updated. To support this you can lock a state. Just right click on it and choose "Locked".



You can also lock and unlock all states via the States menu:

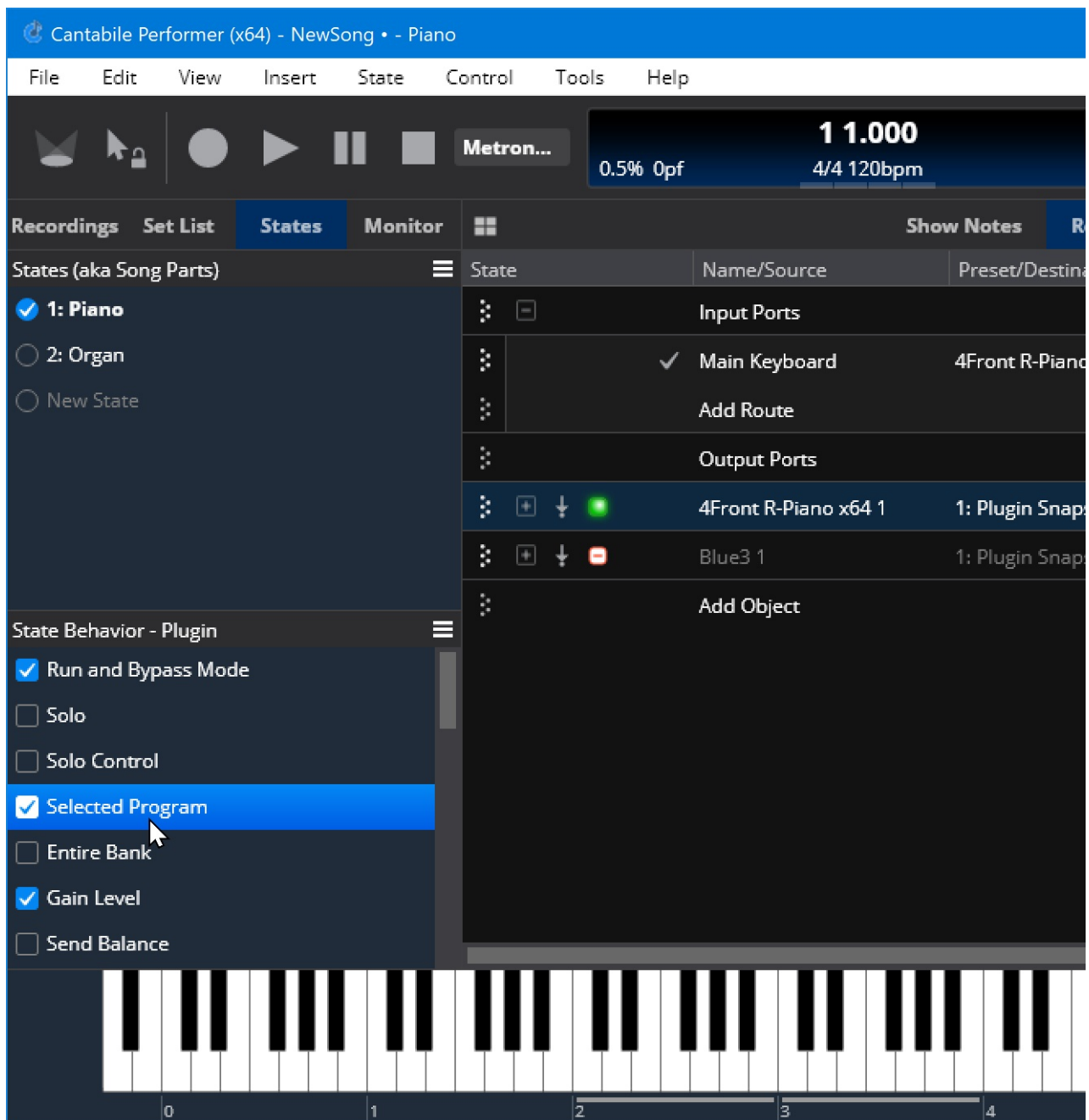


Locking states can also make state switching faster since the current state doesn't need to be re-saved each time.

## State Behaviors

Sometimes there will be attributes that you don't want controlled by states. For example, you might have a MIDI route that should always route to a particular target and you don't want it accidentally changed when switching states. This type of control is called State Behaviors.

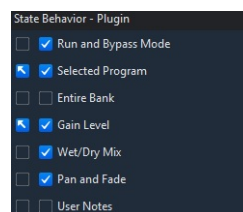
State behaviors are configured in the State Behavior panel which appears below the state list. If hidden you can show it by choosing States Behavior from the View menu.



To change the state behavior of an item, first select it (eg: select a route, plugin, metronome etc...) and then check or uncheck the attributes you want controlled by states.

## Exported Settings

When editing state behaviours in a rack, the panel shows an extra check mark option next to each behaviour.



- Select the first checkmark to indicate that the setting should be exported to the parent song.
- Select the second checkmark to indicate that the setting should be controlled by the rack state.

There are couple of nuances to these settings:

- If only the first option is selected, changing the selected state in the rack will not affect the setting, but switching parent

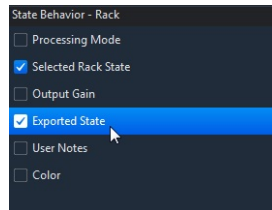
songs (or possibly parent song states) will update the setting.

- If only the second option is selected, switching songs or song states will not affect the setting, but changing the selected rack state will.
- If both options are selected, the setting will be stored in both the rack state and the parent song. Changing the selected rack state will reset the parent song's saved settings to the settings saved in the rack's state.
- If neither option is selected, the setting is not affected by song or rack state changes.

By default, the parent song file will store a copy of each exported setting for each of its song states. ie: each part in the song can individually control the exported settings.

You can change this behaviour with the "Exported State" behaviour on the parent rack slot:

1. Exit from editing the rack
2. Select the rack slot in the parent song file
3. Open the State Behaviour panel
4. Remove the checkmark from the "Exported State" item



Now all settings exported from the rack will be stored on a per-song, rather than a per-song-state basis.

Exported settings allow each song to adjust the setting and reduces the need to create multiple states in the rack. Exported settings can often eliminate the need for rack states entirely.

## Toggling Multiple Behaviours

Sometimes you might like to turn many state behaviours on or off and doing so one at a time would be tedious.

To toggle multiple behaviours on/off all at once, first select the behaviours to updated (using Shift or Control Click or Shift + arrow keys) and press Enter.

To toggle the exported state using Control+Enter.

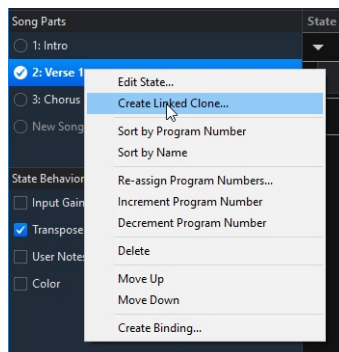
## Linked States

Sometimes it can be useful to create two or more identical states. For example if your song has multiple choruses or verses you might like to create a set of states to represent the sequence of song parts, but each verse and each chorus might be the same.

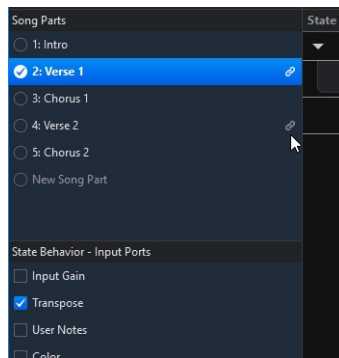
To support this, Cantabile provides "linked states". Linked states each have their own name and program number, but the underlying saved state is shared between them.

If you make changes to a state, any linked states are also updated.

To create a linked state, right click on the original state in the states panel and choose "Create Linked Clone".



Linked states are indicated by a small link icon each state that is linked to the currently loaded state.



In this example, "Verse 2" is a clone of "Verse 1" as shown by the link icons. The "Chorus 2" state is also a clone of "Chorus 1" and would be indicated when either of the chorus states were selected.

## Non-Linked Behaviours

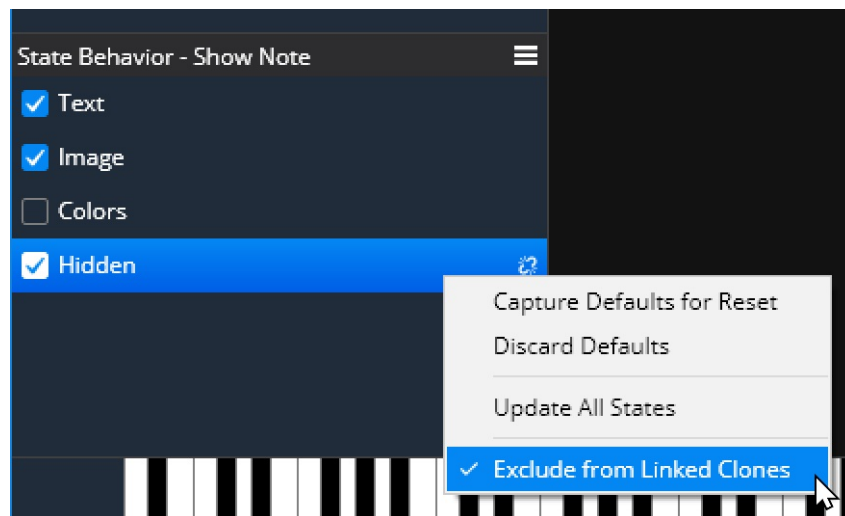
Occasionally you might want to have states where most of the settings are identical, but just one or two settings different.

eg: in the above example linked states were created such that "Verse 1" and "Verse 2" has identical settings. Usually you'll want the same sounds for each verse, but you might want different show notes displayed for each (ie: different lyrics shown for each verse).

In this case you can unlink just the show notes so that each state will have different settings for the show notes, but everything else will be shared.

To unlink a behaviour:

1. Select the object (eg: a show note)
2. Right click on the State Behaviour
3. Choose "Exclude from Linked Clones"



Once selected, those settings won't be shared across linked clone states and each linked clone will maintain its own settings for the selected behaviour.

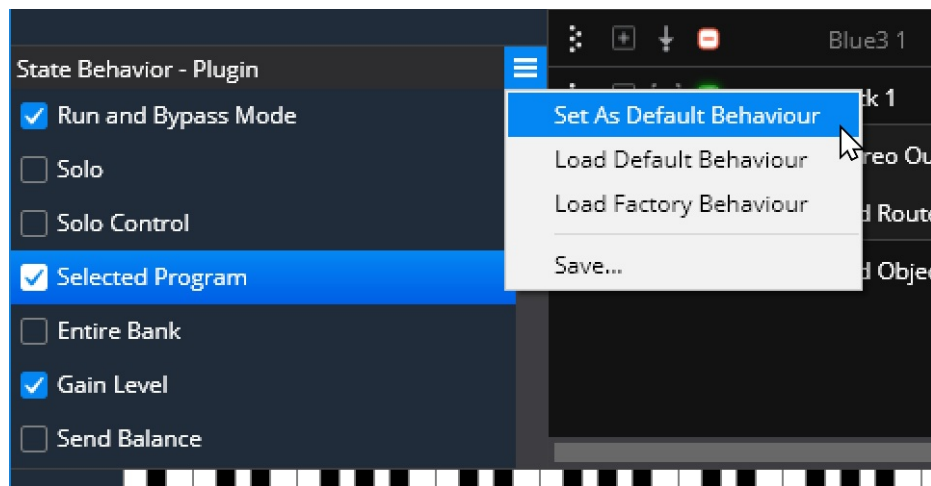
Note that this Exclude setting is also captured by the default state behaviours so if, for example, you want all show notes to be excluded from the linked clones you can configure a note with those exclusions and then capture the default state behaviour as described below.

## Changing the Default State Behaviours

Each kind of state controllable object has a default set of state behaviours that are enabled for new instances of those objects.

For example, each time you add a new plugin to a song it has the following state behaviours enabled: Run and Bypass Mode, Selected Program, Gain Level, Wet/Dry Mix and Pan and Fade.

You can change which behaviours are enabled on new objects by configuring an existing object the way you want the defaults and then choosing "Set As Default Behaviour" from the State Behaviours panel menu:



In the above example, from now on all new plugin instances will only have the Run and Bypass Mode and the Gain Level behaviours enabled.

Note that for plugins, you can't configure the default state behaviour for the plugin specific parameters (since all plugins have different parameters). ie: You can only control the default state of the non-numbered behaviours at the top of the list.

## Lots more

This quick walk-through has only really skimmed the surface of what can be done with states. There are many attributes that can be controlled by states. For example:

- Try selecting different plugin presets in different states
- The transpose and keyboard split settings of a MIDI route can be controlled by states
- States can control the metronome's tempo and time signature
- You can completely unload a plugin in some states if they're not needed. (Right click the plugin slot and set it's run state to unloaded).

The main thing that states can't control is the existence of objects. ie: all states have the same set of plugins, routes etc... eg: if you insert or delete a plugin, it affects all states - not just the active one.

To check which attributes of an object can be controlled by states select the object and check the State Behaviours panel.

## Other Editing Commands

Other useful state editing commands include:

- Press Ctrl+Up/Down or use drag/drop to re-order list items
- Press Delete or use the Edit|Delete menu command to delete states
- Press F2 or use the Edit|Edit State menu command to rename a state
- Undo and Redo commands also work with state editing.

## Set Lists

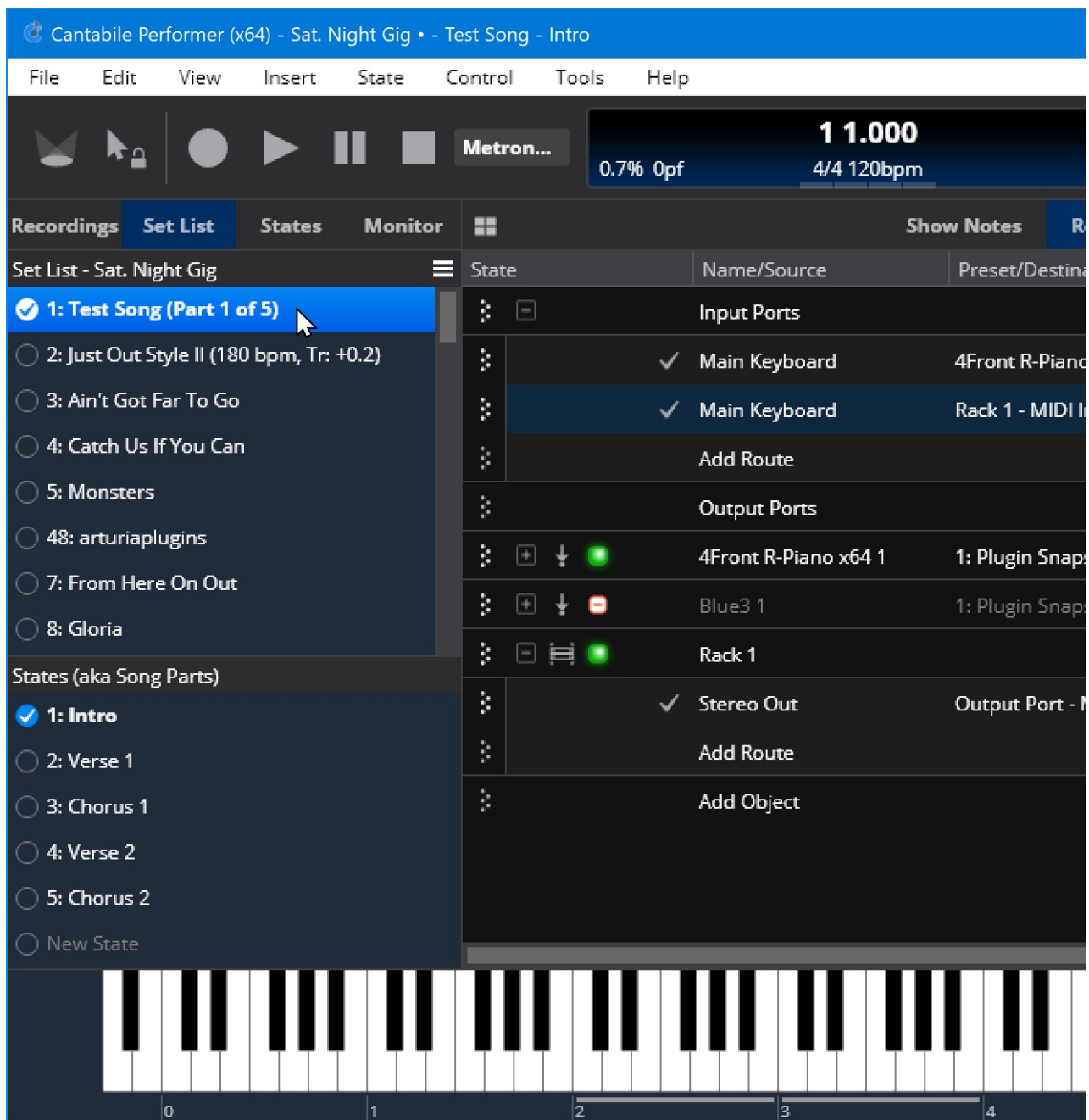
*Cantabile Solo and Cantabile Performer Only.*

A set list is a saved list of song files that can be quickly switched between. You'll typically use a set list to pre-configure the a set of songs in the correct order for a gig.

Each entry in the list stores the name of the song file and an optional program number that can be used to load the song via [Bindings](#).

## Showing the Set List Panel

To work with set lists you use the Set List Panel. To show this panel, either choose View|Set List or press Ctrl+L.

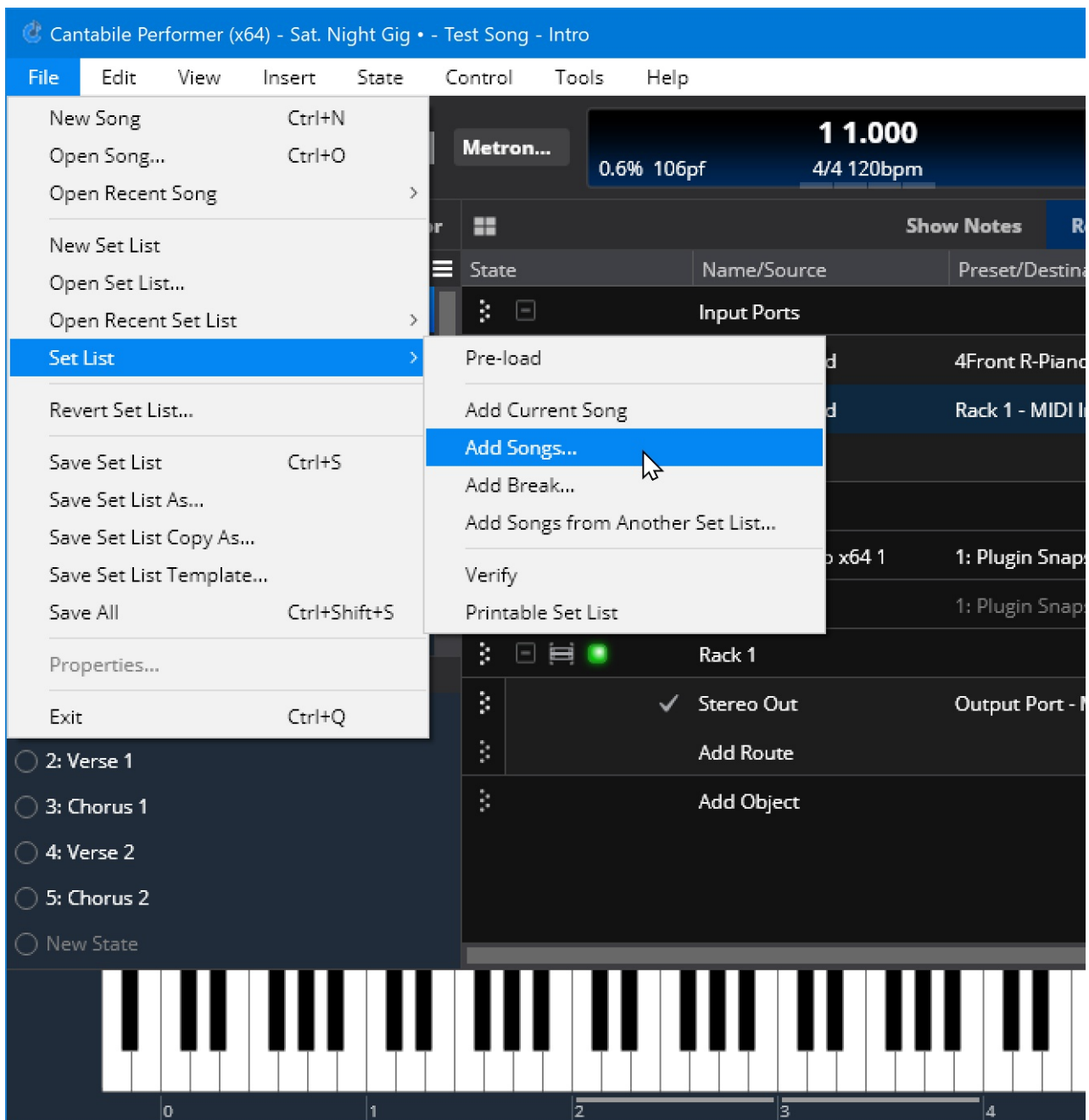


You can hide the set list by resizing it down until it disappears, or by activating it and pressing Shift+Esc.

## Adding Songs to the Set List

To add entries to the set list, from the File menu, choose Set List and then one of the "Add" menu commands.





**Add Current Song to Set List**

Adds the currently loaded song file the set list. Disabled if the current song hasn't been saved.

**Add Songs to Set List**

Brings up a list of all known song files so you can quickly add as many as you like in one hit.

**Add Break to Set List**

A break is a heading within the set list to help you organize the list. eg: you might break your set list into "Early Session" and "Late Session"

**Import Songs from Another Set List**

Imports all the songs from another set list, optionally keeping the associated program numbers (so hard code program number → song bindings will continue to work)

Note: the set list menu commands are also available from the set list panel menu:

## Editing Songs

You can make changes to a song by right clicking it and choosing Edit Song or by selecting it and pressing F2.

Each song has the following settings:

- Song File - the song file to load
- Program Number - a MIDI program number that can be use with Bindings to load specific songs.

Other ways to work with songs include:

- Press Ctrl+Up/Down arrows or use drag/drop to re-order songs (you can select multiple songs and move them all with these commands)
- Clipboard commands can be used to duplicate, re-order and move songs between set lists.
- Set lists support undo and redo.

## Loading Songs

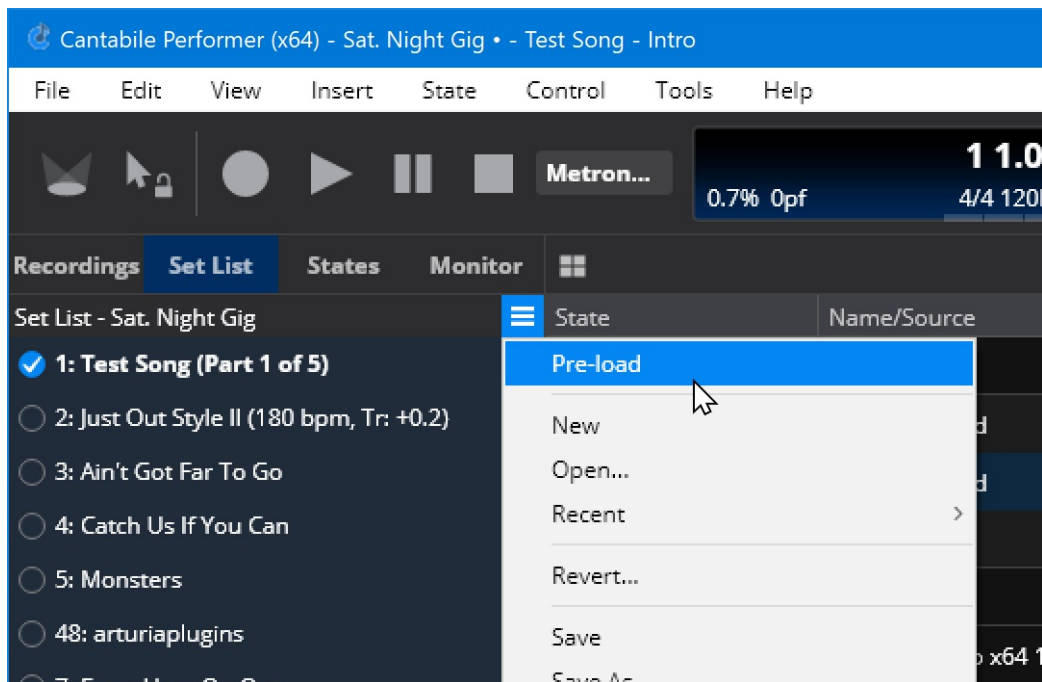
Once you've created a set list you can load songs by:

- Double clicking a song in the list
- Clicking the little circle next to the song's name
- Using [Bindings](#)

## Pre-loaded Set Lists

*Cantabile Performer Only.*

In order to improve song switch times you can instruct Cantabile to pre-load all songs and racks used in the set list:



The pre-load setting is stored as a property of the set list, so you can have some set lists pre-load and some load on demand.

Depending on the plugins you're using, pre-loading a set list may require considerable RAM. You can mitigate this by sharing plugin instances across multiple songs by loading them into racks and using the same rack in each song.

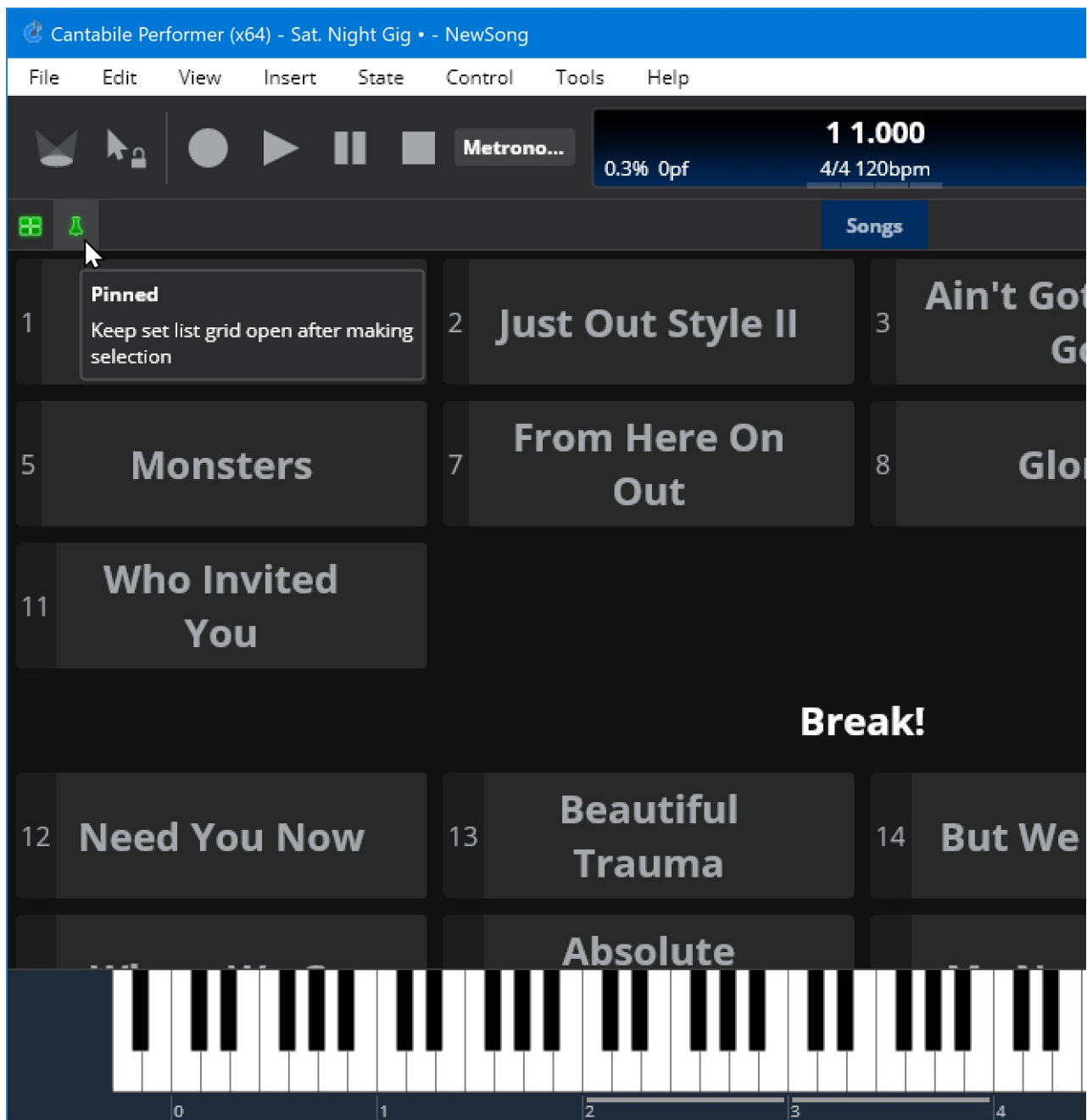
## File Management

Set lists are stored in files with the .cantabileList file extension. Working with set lists is similar to working with other file types and you'll find commands to create, open, save set lists in the File → Set List menu.

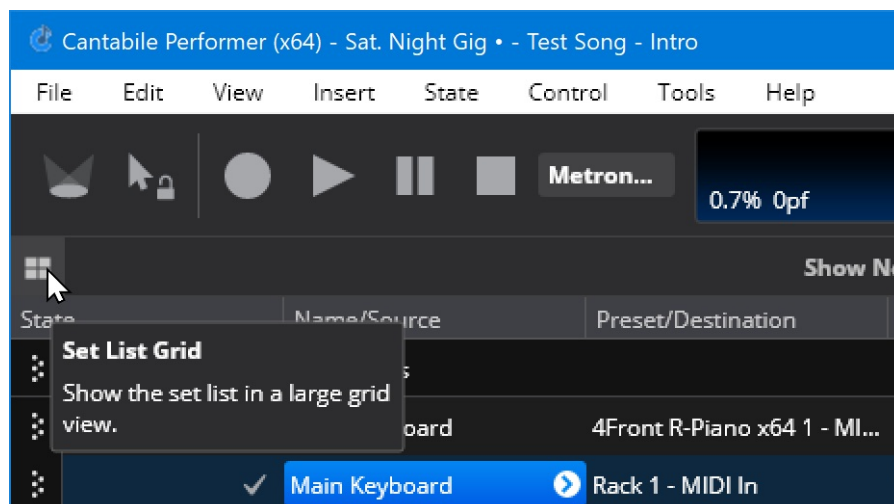
Note that Cantabile stores relative paths between the set list file and the song files that it refers to. If you save your set lists and song files in the same directory, you can move the entire directory and the set list will still be able to locate the relevant song files.

## Set List Grid

The set list can also be displayed in a full-screen style grid:



To show the set list grid, just click the set list grid button:

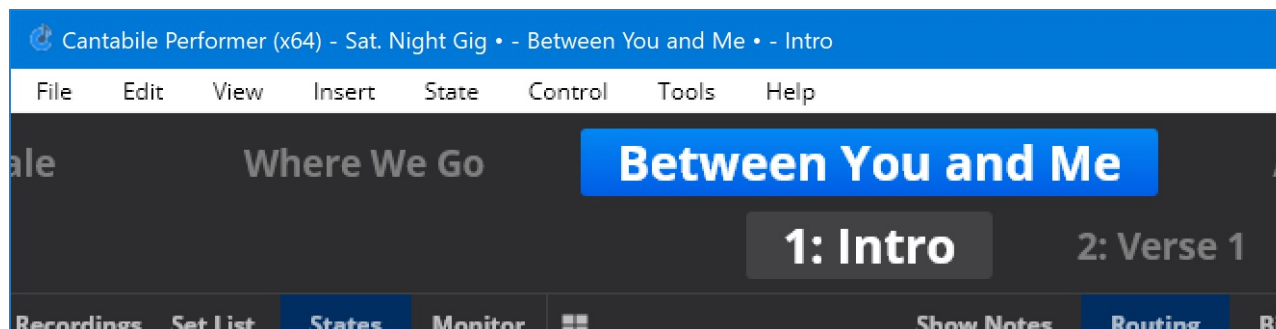


When the set list grid is shown, and small pin icon appears next to the set list grid button. When enabled, the set list grid will stay visible even when switching between songs.

## Ticker Bar

*Cantabile Performer Only*

The Ticker Bar displays two horizontally scrolling lists showing the currently loaded song and song part. Upcoming songs and parts are displayed to the right of the currently loaded item and provide a clear view of your current place in the show.



To show the Ticker Bar, select "View" menu → "Ticker Bar", or press the slash ('/') key

As well as providing a clear view on the currently loaded song and song part, the Ticker Bar can also be used to switch songs and parts:

- With the mouse, click on the name of an item to load it.
- With the keyboard, use the arrow keys to navigate to a song or part and press Enter to load it.

## Customizing the Ticker Bar's Appearance

You can control the size of the text in the Ticker Bar:

1. Move keyboard focus to either the song or song part row
2. Use Page Up and Down to adjust the text size

(Note you can only adjust the size of the loaded item - the items to the left and right will always be displayed at a slightly smaller size than the active item)

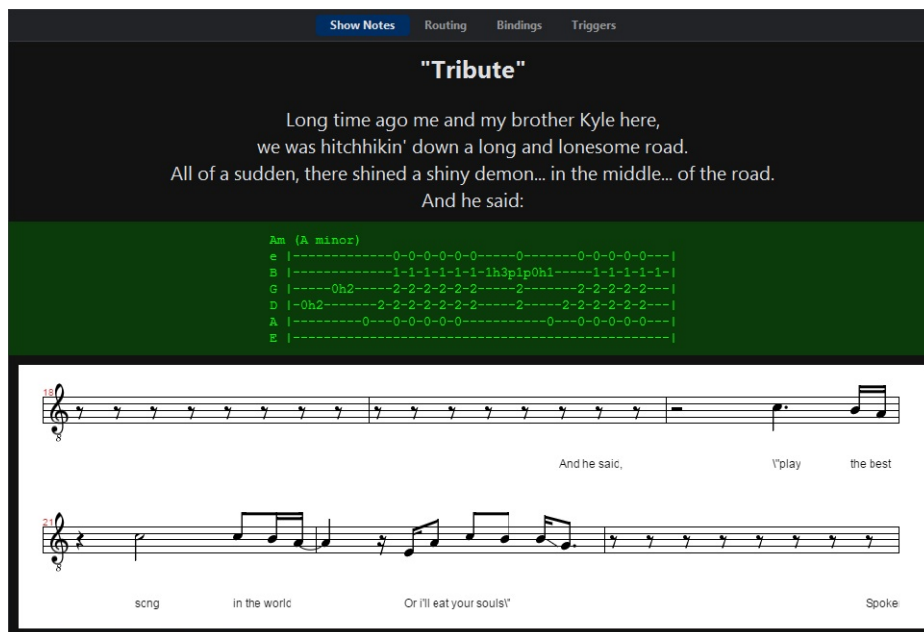
If you don't use song parts, or don't need them displayed so prominently you can hide the second row of the Ticker Bar:

1. Right click anywhere on the Ticker Bar.
2. Choose "Show Song States"

## Show Notes

*Cantabile Performer Only*

The Show Notes tab can be used to store general purpose performance notes such as reminders, lyrics, guitar tabs, images etc...



Show notes are a list of note items where each item comprises a block of text and an image. In the above screen shot the display is comprised of 4 notes:

- The "Tribute" title (Text)
- The lyrics (Text)
- The "ascii art" guitar tab (Text)
- The score display (Image)

## Editing Notes

To add a new note, select "Show Note" from the "Insert" menu, or right click on the Show Notes panel and choose "Add Note". You can edit an existing note by right clicking and choosing "Edit", by double clicking it or by selecting it and pressing enter.

Edit Show Note

Text:

Am (A minor)  
e |-----0-0-0-0-0-0-----0-----0-0-0-0-0-0---|  
B |-----1-1-1-1-1-1-1h3p1p0h1-----1-1-1-1-1-1-|  
G |----0h2-----2-2-2-2-2-2-----2-----2-2-2-2-2---|  
D |-0h2-----2-2-2-2-2-2-----2-----2-2-2-2-2-2---|  
A |-----0---0-0-0-0-0-0-----0---0-0-0-0-0-0---|  
E |-----|

Text Alignment: Center Text Size: 16 ☐ Bold ☒ Fixed Pitch

Background Image:

☐ Hidden

OK

Most of these properties are self explanatory, except...

- Images are always displayed centered and reduced in size if too wide to fit on screen.
- Fixed pitch text items are aligned as a block (rather than each line individually aligned).
- Notes also have colors which can be selected by right clicking on the item on the main Show Notes panel.

Notes can be cut, copied, pasted and re-ordered using the standard short-cut keys and menu commands.

## Resizing Images

Images used in show notes can be resized by selecting the show note (indicated by blue border) and using the Ctrl+Page Up and Ctrl+Page Down keys to adjust the size.

## Hidden Items

Notes can be hidden to remove them from view (typically used in combination with states - see below).

In order to access hidden notes for editing, right click on the Show Notes panel and select "Show Hidden Notes". The hidden notes will be shown in washed out colors.

## Variables

You can use variables to include dynamic text in the text area of a show note.

eg: to include the name of the current song in a show note:

Now Playing: \$(SongTitle)

For a full list, see [variables](#).

## Controlling Notes with States

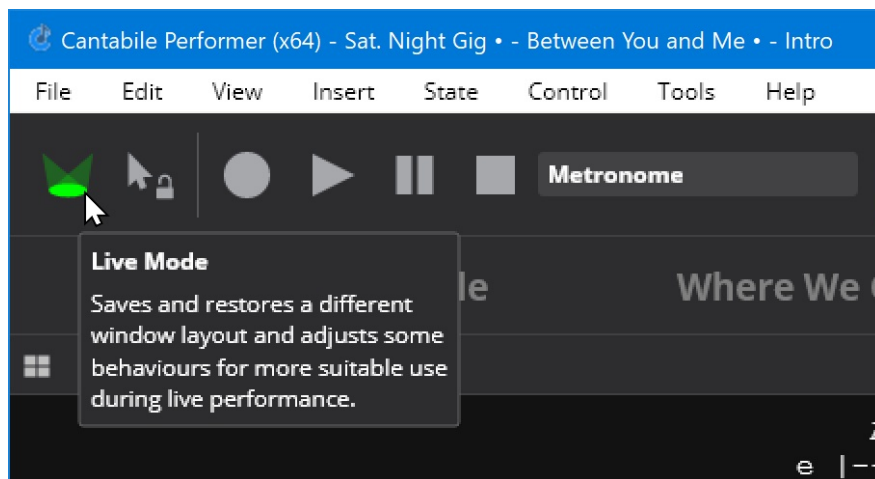
Show Notes can be controlled by song states so different song parts can show different notes.

The attributes of the note that are controlled by states can be selected in the State Behaviour panel just like any other state controllable object.

See also: [States](#)

## Live Mode

Cantabile's Live Mode button lets you maintain two separate layouts of the main window - one for editing songs and racks and one for during live performance.



To toggle between Normal Mode and Live Mode, click the Live Mode button (see above screen shot) or choose "Live Mode" from the "View" menu or press the F5 key.

Note that Live Mode has no other side effects other than changing the appearance and layout of Cantabile's main window. You can still switch to other tabs and activate other panels during Live Mode.

## Configuring Live Mode

By default Live Mode is configured to show user interface elements useful during live performance. eg: the monitor panel, controller bar, ticker bar and show notes panels are shown while the on-screen keyboard, media file time line, and metronome panels are hidden.

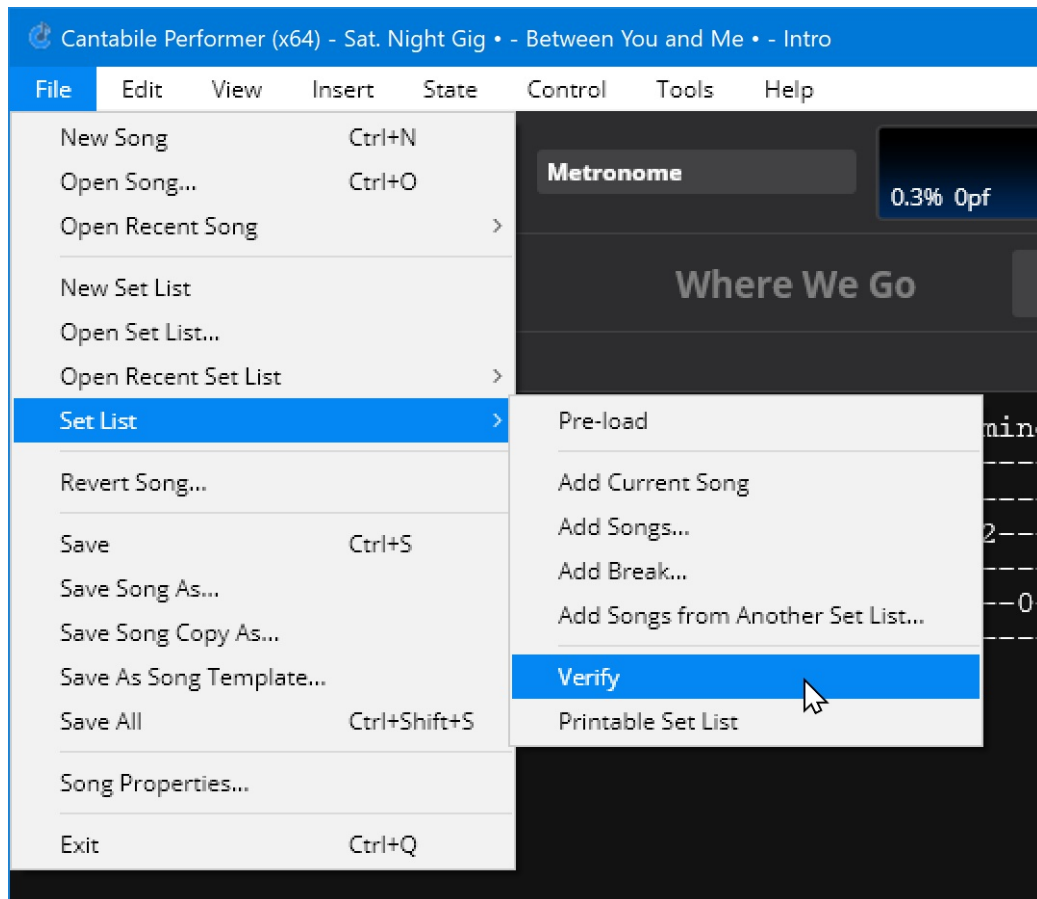
You can configure the layout of each mode and Cantabile will remember that layout when you switch back to that mode. For example, suppose you want the media file time line panel visible during live performance:

1. Activate Live Mode
2. Activate the timeline panel (Ctrl+E)
3. Deactivate Live Mode

Now every time you switch to Live Mode the timeline panel will be shown.

## Set List Verification and Printing

Cantabile includes a couple of tools for working with set lists:



### Pre-load Set List

Pre-loads all the songs in the set list and all referenced racks and plugins. See [Set Lists](#) for more.

### Verify Set List

Checks the set list and all referenced song files for errors including missing plugins, racks, media files etc... and generates a report. [See here](#) for more details.

### Printable Set List

Generates a HTML version of the set list suitable for printing and opens it in a web browser. [See here](#) for more including how to customize the output.

## Preventing Prompts to Save

By default Cantabile will prompt to save changes to a songs, racks and set lists before closing them. Generally this is a good idea however during live performance, such prompts are completely unwanted.

For this reason, Cantabile provides some options to control when these prompts are displayed. See Options → General → Save And Load:

Save and Load	
Save Songs when switching via File menu:	Prompt
Save Songs when switching via MIDI/Network:	No
Save Songs when switching via Set List:	No
Save Set Lists:	Prompt

Each option has the following choices:

Yes

The file will be saved without prompting - changes will be preserved.

No

The file will never be saved - changes will be lost.

Prompt

You'll be prompted whether to save or not (and given the chance to cancel)

See also [Song and Rack Options](#) for options to control when a file is marked as modified.

## Command Line Options

Cantabile supports the command line options listed below.

Note that if Cantabile is already running, most of these commands will be ignored and instead the existing Cantabile window will be activated.

*fileName*

Name of the file to open. Can be a .cantabileSong, .cantabileSetList or .cantabileLicense file

*/driver:driverName*

Sets the audio driver to use, overriding and replacing the one selecting in settings. See [blog post](#).

*/config:configname*

The name of the configuration to use. See [Multiple Configurations](#)

*/state:stateName*

When loading a song also specified on the command line, load it with this state

*/livemode*

Start up in with Live Mode turned on

*/headless*

Disable various prompts and dialogs that might interfere with running on a headless machine setup

*/dontstartengine[:milliseconds]*

Don't automatically start the audio engine. Cantabile will not process audio/MIDI until the engine is manually started (via the power button top right of main window, or Tools Menu → Run Audio Engine). Starting with build 3679 an optional delay milliseconds can be included eg: `/dontstartengine:5000` will delay starting the audio engine for 5 seconds.

*/minimized*

Run with the main window minimized

*/maximized*

Run with the main window maximized

*/regserver*

Create shell association for Cantabile files

*/unregserver*

Remove shell associations for Cantabile files

## Controller Defaults

Cantabile supports various ways of mapping controllers (typically MIDI CC controllers) to binding targets. For example a CC may be interpreted as a regular continuous controller, a button, a switch, a rotary encoder etc... Normally you configure how a controller is interpreted as part of configuring the binding however you can also create a file that defines how these bindings should be created automatically.

This page describes how to configure these controller defaults and should be considered an advanced topic that most users shouldn't need to bother with.

*This feature requires Cantabile build 3624 or later*

### controllerDefaults.json

Controller defaults are configured in a special file called "controllerDefaults.json" which should be placed in Cantabile's [settings folder](#) and must be a valid [JSON](#) file.



The controller defaults file should have the following structure:

```
{
  "controllerDefaults":
  [
    {
      // settings for controller defaults 1
    },
    {
      // repeat as many times as necessary
    },
  ]
}
```

## Match Settings

Within the inner-most braces of the above section the following settings can be configured and specify the kind of incoming MIDI event this controller default applies to.

### portName

The port name the controller is received from. This must exactly match (case sensitive) the MIDI port name as configured in Cantabile Options → MIDI Ports

### sourceMidiEventKind

The MIDI Event kind that these defaults apply to and should be one of the following values.

Note that if this is set Controller, ControllerButton, ControllerNonEdgeButton or ControllerSwitch then this will match any of controller kinds and will cause the binding to acquire the controller type specified in the controller defaults file.

For example suppose you configure a controller default as "ControllerButton" then created a binding of type "ControllerSwitch" then the binding will be forced to "ControllerButton" (assuming all other match settings match).

- Note
- Controller
- ProgramChange
- PitchBend
- ChannelPressure
- NoteOff
- NoteSwitch
- ControllerButton
- ControllerNonEdgeButton
- ControllerSwitch
- FineController
- BankedProgramChange
- RpnCoarse
- RpnFine
- NRpnCoarse
- NRpnFine
- MasterVolume
- MasterBalance
- MmcBase
- MmcStop
- MmcPlay
- MmcDeferredPlay
- MmcFastForward
- MmcRewind
- MmcRecordPunchIn
- MmcRecordPunchOut
- MmcRecordReady
- MmcPause
- MmcEject
- MmcChase
- MmcReset
- TransportPlay
- TransportStop
- TransportPause
- SongSelect
- ClockStart
- ClockContinue
- ClockStop

### sourceControllers

An array of source controller numbers that should be matched against.

## Assignment Settings

If a binding matches all of the above "match settings" then the following settings (all optional) will be applied to the binding to configure it:

### controllerEncoding

The controller encoding mode, one of the following:

- Absolute
- AbsoluteWithJumpPrevention
- RelativeMode1
- RelativeMode2
- RelativeMode3

### relativeScaling

The scaling factor for relative encoders, where 1.0 = 100%

### sourceRangeMin and sourceRangeMax

The source range of the controller

### curveKind and curveAmount

The curve kind and amount to apply to this binding. The curveKind setting can be one of the following:

- None
- ExpandCompress
- Enhance
- InverseExpandCompress
- InverseEnhance

## Example

For example, suppose CC controllers 16, 17, 18 and 19 on your main keyboard are all rotary encoders, creating a controller defaults file similar to the following would cause bindings against those controllers to always be configured as rotary encoder with a scaling of 130%.

```
{
  "controllerDefaults":
  [
    {
      "portName": "Main Keyboard",
      "sourceMidiEventKind": "Controller",
      "sourceControllers": [ 16, 17, 18, 19 ],
      "controllerEncoding": "RelativeMode1",
      "relativeScaling": 1.3
    }
  ]
}
```

## Error Logging

If there are syntax errors in the controllerDefaults.json file, an error will be logged to Cantabile's log file and none of the controller defaults will be loaded nor used.

## Custom Themes

Cantabile supports loading custom themes for the main window. This page describes how to create your own custom theme.

### Default Theme File

The default themes provide a good starting point for creating your own theme. These themes were authored using [Sketch by Bohemian Coding](#) (requires OS-X). Although you can create themes using any tools you like, Sketch really does work well for this and is highly recommended.

The assets for the default themes are [available here](#).

You'll find two sketch files there: Light.sketch and Dark.sketch. You might also like to examine the theme files that ships with Cantabile. They're in the same folder as Cantabile.exe and have a .theme extension. Rename them to have a .zip file extension to see their contents.

## Locating Theme Files

Cantabile locates theme files by looking first in the [resource folder](#) and then in the same folder as the Cantabile executable.

## Theme File Format

A Cantabile theme consists of:

1. a set of graphic resources in .png format.
2. a theme.json file that specifies colors for text and other elements.

## Inheriting from a Base Theme

A theme can inherit most of its assets and settings from a base theme. This makes it easy to tweak just a few assets in an existing similar theme.

The base theme is specified in the theme.json file:

```
{
  "baseTheme": "Dark"
}
```

The value specified should be the plain theme name and not a file name. Cantabile will search its resource path to find the appropriate theme file. (Be careful to avoid creating a circular inheritance).

By inheriting from one of Cantabile's built-in themes you can save having to update your theme every time a new graphic asset it introduced since the built-in themes will always be able to provide those new assets.

The minimum requirements for a theme are a theme.json file with the name of a base file. Everything else can be inherited from that base theme file including colors and strings in theme.json and graphics resource assets.

*Theme inheritance requires build 3500 or later*

## Graphic Resources

Each graphic resource should be provided in three sizes and conform to the following naming convention:

- basename.png - the resource at standard 96 DPI resolution
- basename@2x.png - the resource at double the resolution
- basename@4x.png - the resource at 4x the resolution

The @2x and @4x resolution images are not strictly necessary but will result in blurred rendering on high resolution monitors or when the main window scaling factor is high. (Sketch can automatically render assets at different resolutions and you'll see the supplied sketch files are configured to do this).

The names of the required resource files can be determined by looking at the slice names in the above sketch files.

If a resource can't be found in the theme, Cantabile will use a red and yellow "flag" image to highlight missing theme assets.

## Specifying Text Colors with theme.json

The theme.json file specifies various non-graphic settings related to the theme - mainly text colors. It's format and the contained fields are self explanatory.

## Packaging Theme Files

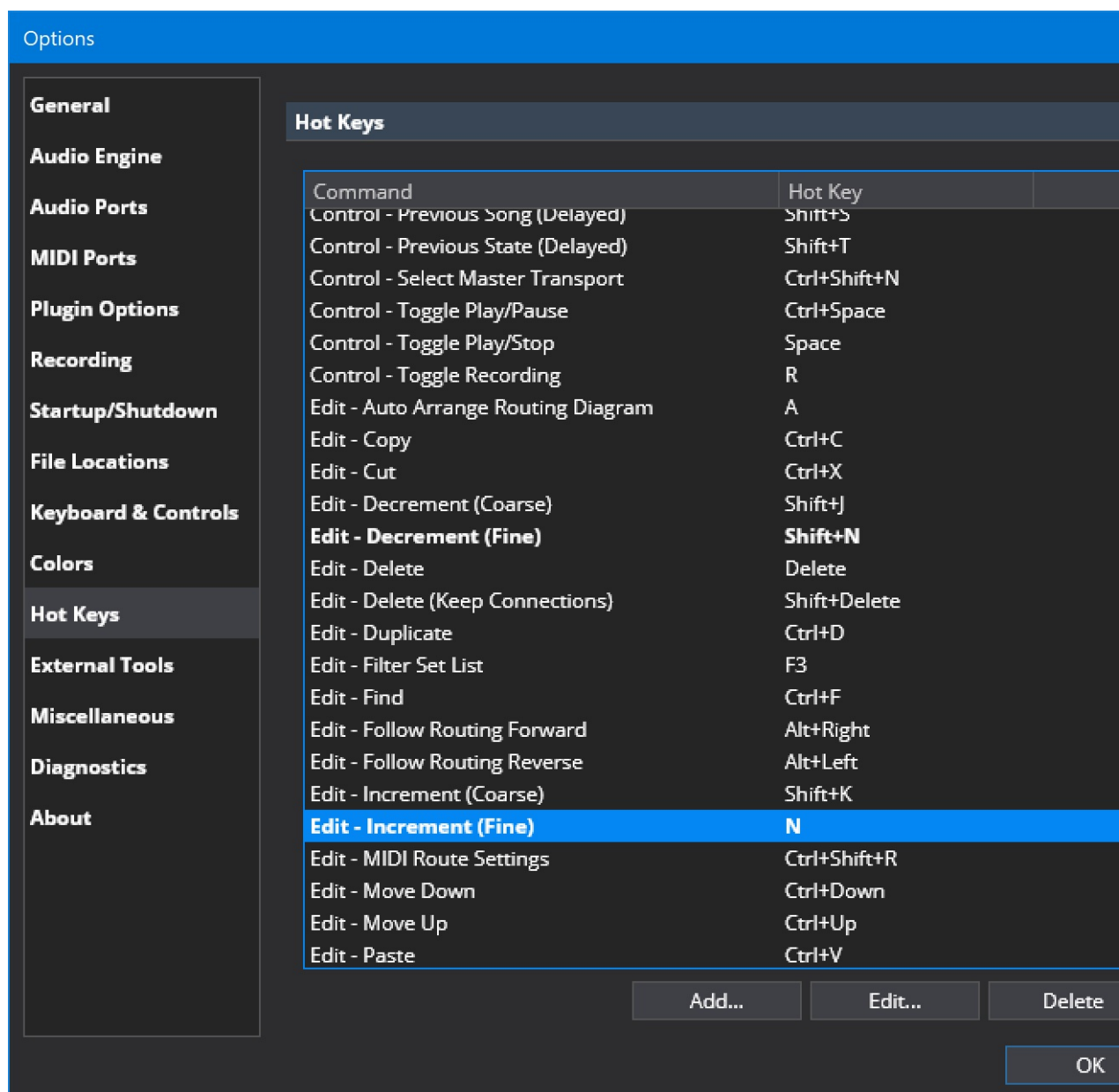
A theme can be either a zip file (renamed to have a .theme file extension) or a folder (also with a .theme extension) containing the above assets.

When packaging theme assets into a zip file compression *should not* be used.

PNG files are already zip compressed and re-compressing a second time only slows down load times. See the buildzips.bat file in the above mentioned repository for how Cantabile's build process packages themes.

## Customizing Shortcut Keys

Cantabile comes with a preconfigured set of shortcut keys for common commands. You can however customize these via the Options → Hot Keys options page:



To customize the keys you can either double click an existing key binding and edit it, or click the Add button and create a new key binding.

Note that there are many commands that are not included in the default key binding set - use the Add button to discover other available commands.

Tip: you'll notice that default key bindings are displayed in normal font while custom short cut keys are displayed in bold.

## Expression Reference

Cantabile includes an expression evaluation engine that is used for evaluating the expressions in [Gain Control Curves](#).

### Data Types

The expression engine only supports double precision floating point numbers.

For boolean operations, any non-zero value is considered true and any function or operator that returns a boolean value will return 1 for true and 0 for false.

### Functions

Functions are called by name with parameters enclosed in round brackets and separated by commas.

eg: `pow(10, 3)`

Functions are defined with a function prototype as the JSON key and the value the expression:

eg: `"mySillyAddFunction(a,b)": "a+b"`

## Constants

Constants are functions with no parameters and may be defined as JSON numeric value:

eg: `"curveSlope": 60`

or as an expression:

eg: `"curveSlope": "120/2"`

## Operators

The expression engine supports the following operators:

- `+` Add
- `-` Subtract and negative
- `*` Multiply
- `/` Divider
- `? :` Ternary (aka conditional) operator
- `<, <=, >, >=, ==, !=` Comparison
- `!` Logical Not
- `&&` Logical And
- `||` Logical Or
- `( )` Grouping

## Built-in Functions

The expression engine includes the following built-in functions. The unit for all angles is radians.

- `abs(x)` Absolute value
- `acos(x)` Arc-cosine
- `asin(x)` Arc-sine
- `atan(x)` Arc-tangent
- `atan2(x,y)` Arc-tangent of (x,y)
- `ceiling(x)` Round to more positive value
- `cos(x)` Cosine
- `cosh(x)` Hyperbolic Cosine
- `e` Constant e
- `exp(x)` e raised to the power of x
- `floor(x)` Round to less positive value
- `log(x)` Natural (base e) logarithm
- `log(x, b)` Base b logarithm
- `log10(x)` Base 10 logarithm
- `min(x,y)` The smaller value of x and y

`max(x,y)`  
 The larger value of x and y  
`pi`  
 Constant pi  
`pow(x, y)`  
 x raised to the power y  
`round(x)`  
 Round x to the closest whole enumber  
`sign(x)`  
 Returns -1, 0, or 1 matching the sign of x  
`sin(x)`  
 Sine  
`sinh(x)`  
 Hyperbolic Sine  
`sqrt(x)`  
 The square root of 'x'  
`tan(x)`  
 Tangent  
`tanh(x)`  
 Hyperbolic Tangent  
`truncate(x)`  
 Remove the fractional part of a x  
`isnan(x)`  
 Returns 1 if x is not a number, else 0  
`isinfinity(x)`  
 returns 1 if x is infinity, else 0  
`infinity`  
 The constant +infinity  
`nan`  
 The constant for non a number  
`epsilon`  
 The smallest expressible double precision number  
`maxDouble`  
 The most positive supported double precision number  
`minDouble`  
 The most negative supported double precision number  
`epsilonFloat`  
 The smallest expressible single precision number  
`maxFloat`  
 The most positive supported single precision number  
`minFloat`  
 The most negative supported single precision number  
`toDb(x)`  
 Converts a scalar value to decibels  
`toDb(x,y)`  
 Converts a scalar value x to decibels with a floor of y dB. (ie: a smaller value will be returned as negative infinity)  
`fromDb(x)`  
 Converts decibels to scalar  
`fromDb(x, y)`  
 Converts decibels to scalar with a floor of y dB (ie: a smaller value will be returned as 0);

## Functions

Cantabile's expression engine supports the following functions. See also the [Variable Reference](#) and [Sys-ex Expressions](#).

### Math Functions

.NET Math functions  
 All the static functions of the [.NET Math class](#)  
`clamp(value,min,max)`  
 Clamps a value between a specified range  
`hi7(x)`  
 returns the high-7 bits of a 14-bit number  
`hibyte(x)`  
 returns the high-8 bits of a 16-bit number  
`lo7(x)`  
 returns the low-7 bits of a 14-bit number  
`lobyte(x)`  
 returns the low-8 bits of a 16-bit number  
`int(x), uint(x), long(x), ulong(x), short(x), ushort(x), byte(x), sbyte(x)`  
 converts a value to a specified integer type (typically used for forcing a floating point value to integer)

### Miscellaneous Functions

`length(x)`  
 returns the number of elements in an array, the number of characters in a string or 1 for other types.

## String Functions

ASCII(str)

ASCII-7 encodes a string, returning a byte array

formatNumber(value, formatString)

Formats a number using a [C# double format specifier](#). eg: formatNumber(123.123456, "N2") to format to 2 decimal places.

EndsWith(a, b)

Checks if string a ends with b. (Case insensitive)

StartsWith(a, b)

Checks if string a starts with b. (Case insensitive)

strcat(a,b)

joins two strings together, although the + operator can also be used to join strings.

strhas(str, find)

returns true if the string find is contained in str, otherwise false. The comparison is case-insensitive and culture invariant.

strmatch(str, regex)

returns true if the string str matches the regular expression regex. The comparison is case-insensitive and culture invariant.

strpart(str, i)

splits a string on semicolon ; characters and returns the ith part, or an empty string if not enough parts are available.

eg: strpart("Apples;Pears;Bananas", 1) returns "Pears".

strstr(str, find)

returns the zero-based index of the first occurrence of find in str, or -1 if not found. The comparison is case-insensitive and culture invariant.

substr(str,pos)

returns a sub-string of str starting at character pos. If pos is negative, returns the end of the string

substr(str,pos,length)

returns a length characters from str starting at character pos. If pos is negative, returns the end of the string

UTF7(str)

UTF-7 encodes a string, returning a byte array

UTF8(str)

UTF-8 encodes a string, returning a byte array

UTF16(str)

UTF-16 (little endian) encodes a string, returning a byte array

UTF16\_be(str)

UTF-16 (big endian) encodes a string, returning a byte array

## Velocity and Control Curve Functions

CCToGain(value)

Maps a coarse controller value (0-127) to a gain scalar using the user-selected MIDI gain control curve.  
curve(curveKind, curveAmount, value)

Converts a value using one of Cantabile's built-in a velocity curves.

- curveKind: one of "enhance", "expand", "enhance-1", "expand-1"
- curveAmount: the curve amount from -1.0 to 1.0 (and where 0.0 = linear mapping)
- value: a position on the curve from 0.0 to 1.0
- returns: the curve value at the position from 0.0 to 1.0

fineCCToGain(value)

Maps a fine controller value (0-16383) to a gain scalar using the user-selected MIDI gain control curve.

gainToFineCC(value)

Maps a gain scalar to a fine controller value (0-16383) using the user-selected MIDI gain control curve.

gainToCC(value)

Maps a gain scalar to a coarse controller value (0-127) using the user-selected MIDI gain control curve.

positionToScalar(controlCurveName, value)

Converts a control curve position to a scalar gain value using one of the user-selected control curves.

- controlCurveName: "gain", "midi" or "levels"
- value: a control curve position from 0.0 to 1.0
- returns: a scalar gain value (ie: where 1.0 = 0dB)

scalarToPosition(controlCurveName, value)

Converts a a scalar gain value to a control curve position using one of the user-selected control curves.

- controlCurveName: "gain", "midi" or "levels"
- value: a scalar value to be converted
- returns: a control curve position from 0.0 to 1.0

## Decibel Functions

formatDb(value [, muteString])

Formats a decibel value to a string. If less than approx -60b returns  $-\infty$  or the muteString if supplied.  
 formatGain(value [, muteString])  
 Formats a scalar gain value to decibels string. If less than approx -60b returns  $-\infty$  or the muteString if supplied.  
 fromDb(value [, limit])  
 Converts a decibel value to scalar. If limit specified and value < limit returns zero.  
 toDb(value [, limit])  
 Converts a scalar gain value to decibels. If limit specified and value < limit returns negative infinity.

## MIDI Functions

formatMidiNote(value)  
 Formats a MIDI note number using Cantabile current settings for display note names.  
 formatProgramNumber(value)  
 Formats a program number according to current Cantabile settings for displaying program numbers.  
 midivarlen(x)  
 encodes a multi-byte integer using 7-bit MIDI variable length encoding.  
 parseProgramNumber(string)  
 parses a Cantabile formatted banked program number returning a zero based banked program number.  
 parseMidiNote(string)  
 parses a Cantabile formatted MIDI note name (eg: "C#4") and returns MIDI note number.

## Onscreen Keyboard State

These functions return the state of the onscreen keyboard. In all cases the MIDI channel number is optional and defaults to 1 if not specified.

is\_note\_held(noteNumber, [channel])  
 Returns 1 if the note is held, otherwise 0  
 lowest\_held\_note([channel])  
 Returns the MIDI note number of the lowest held note, or -1 if no held notes  
 lowest\_held\_pitch\_class([channel])  
 Returns the pitch class as a string (eg: "C", "C#" etc...) of the lowest held note  
 all\_notes\_held(notesArray, [channel])  
 Returns 1 if all notes in the array are held, otherwise 0 eg: all\_notes\_held([60,64,67], 2) checks if all notes in middle CMaj on MIDI channel 2 is held  
 any\_notes\_held(notesArray, [channel])  
 Returns 1 if any notes in the array are held, otherwise 0 eg: any\_notes\_held([60,64,67], 2) checks if any notes in middle CMaj on MIDI channel 2 are held  
 is\_pitch\_class\_held(pitch\_class, [channel])  
 Returns 1 if at all specified pitch classes are held. eg: is\_pitch\_class\_held("CEG") returns 1 if at any C, E and G notes are held (ie: all three must be held, but in any octave)  
 any\_pitch\_class\_held(pitch\_class, [channel])  
 Returns 1 if at any of the specified pitch classes are held. eg: any\_pitch\_class\_held("CEG") returns 1 if any C, E or G notes are held  
 cc(controllerNumber, [channel])  
 The current value of a continuous controller  
 finecc(controllerNumber, [channel])  
 The current value of a fine continuous controller  
 program([channel])  
 The currently selected program number  
 bankedProgram([channel])  
 The currently selected banked program number  
 pitchBend([channel])  
 The current pitch bend value  
 channelPressure([channel])  
 The current channel pressure value  
 rpnCoarse(paramNumber, [channel])  
 The current coarse value of an RPN parameter  
 rpnFine(paramNumber, [channel])  
 The current fine value of an RPN parameter  
 nrpnCoarse(paramNumber, [channel])  
 The current coarse value of a NRPN Parameter  
 nrpnFine(paramNumber, [channel])  
 The current fine value of a NRPN Parameter

## Roland Sys-ex Functions

RolandChecksum(array)  
 calculates the Roland sys-ex checksum of an array. (The array is flattened to a byte array using the same rules as for [sys-ex expressions](#))  
 RolandPack16(value)  
 packs a 16-bit value into 4 bytes with 4-bits in each byte (MSB first), as used in Roland sys-ex. (aaaabbbbccccdddd => 0000aaaa 0000bbbb 0000cccc 0000dddd)



RolandPack8(value)  
packs an 8-bit value into 2 bytes with 4-bits in each byte (MSB first), as used in Roland sys-ex. (aaaabbbb => 0000aaaa 0000bbbb)

## MIDI Generations Functions

These functions are only available in the "Sys-Ex / Raw Bytes" binding point target and can be used to generate raw MIDI data.

Note the parameters to these functions are all zero-based, except the MIDI channel number which is 1 based.

midi.noteOn(channel, noteNumber, velocity)  
generates the 3-bytes of a note on event.  
midi.noteOff(channel, noteNumber)  
generates a 3-byte note on event with velocity zero (ie: a note off event)  
midi.cc(channel, controllerNumber, controllerValue)  
generates a standard 3-byte continuous controller event.  
midi.fine\_cc(channel, controllerNumber, controllerValue)  
generates a fine-valued (0-16384) continuous controller event  
midi.after\_touch(channel, noteNumber, value)  
generates a 3-byte after-touch event  
midi.program\_change(channel, programNumber)  
generates a 2-byte program change event  
midi.bank\_select(channel, msb, lsb)  
generates a bank select messages as 2x CC events  
midi.banked\_program\_change(channel, programNumber)  
generates a banked program change as 2x CC events and 1x program change event  
midi.rpn(channel, parameterNumber, parameterValue)  
generates a fine-RPN event as series of 4x continuous controller events  
midi.nrpn(channel, parameterNumber, parameterValue)  
generates a fine-NRPN event as series of 4x continuous controller events  
midi.channel\_pressure(channel, pressure)  
generates a 2-byte channel pressure event  
midi.pitch\_bend(channel, value)  
generates a 3-byte pitch bend event  
midi.delay(milliseconds)  
generates a special sys-ex event encoding a delay period. This special sys-ex is intercepted by Cantabile when processing the generated MIDI byte stream and causes a sample accurate delay before the following events are emitted.

## Gain Control Curves

Cantabile supports various Gain Control Curves. The Gain Control Curves determine how Cantabile responds to the movement of a gain slider. Different curves provide different top end gain and different sensitivities at different parts of the curve.

### Selecting Control Curves

In Options → Keyboard and Control there are a three different settings that determine which control curve to use in various situations:



Control Curves	
Slider Gain Control Curve:	Cantabile (New) ▼
Level Meter Control Curve:	Cantabile (New) ▼
MIDI Control Curve:	Cantabile (New) ▼

### Built-in Control Curves

Cantabile includes 4 built-in control curves:

Cantabile (Classic)

The same control curve as older builds of Cantabile which has a range from silent up to +14dB. It also includes a hard coded flat point around 0dB for snapping.



#### Cantabile (Classic MIDI)

The same control curve as older builds of Cantabile for MIDI bindings to control curves. This is a linear curve against a dB scale from -60dB to +14dB.

#### Cantabile (New)

A new curve that provides more precise control and ranges from silent up to 7.3dB with the 0dB level 3/4 of the way along on the slider. When used with MIDI bindings, a CC value of 96 maps to 0dB.



#### Cantabile (0db Center)

The same curve as above but with the 0dB position moved to the center of the slider giving a higher top end gain of about 18dB. A MIDI CC value of 64 maps to 0dB.



## Custom Control Curves

Cantabile also supports custom control curves by authoring a curve definition file.

A curve definition file is a text file that includes math expressions that define how to map between a slider position and an amplitude gain and vice versa. They also include information on where to display slider tick marks and level meter color locations.

Control curve files have a file extension of ".cantabileControlCurve" and the four standard curves can be found in the Cantabile's installation folder. They can also be [viewed here](#).

The control curves are located by looking in two locations.

1. The Resource Folder (see Options → File Locations)
2. The same folder where Cantabile is installed

If a file with the same name is found in both locations, the one in the resource folder is used.

Each file must be a properly formatted [JSON](#) file although Cantabile's JSON parser also allows for Javascript style comments. The file is divided into two main sections "definitions" and "uiKinds".

## "definitions" Section

The definitions section defines a set of functions and constants. It can include as many different definitions as you like however it must include two functions "positionToScalar(x)" and "scalarToPosition(x)", as shown in the following example:

```
"definitions":
{
  // All control curves must provide these two functions
  // (which must be the inverse of each other else weirdness will ensue).

  // - Position ranges from 0.0 → 1.0 and represents the slider position
  // - Scalar is the amplitude scaling (ie: linear multiplier)

  "positionToScalar(x)": "pow(10, (log(x, 10) * slope - (log(zeroDbPos, 10) * slope)) / 20)",
  "scalarToPosition(x)": "pow(10, (log(x, 10) * 20 + (log(zeroDbPos, 10) * slope)) / slope)",

  // Higher value makes the slope of the curve steeper and gives a higher top gain level
  "slope" : 60,

  // Position of the 0 dB mark
  "zeroDbPos": "96/127",
},
```

These functions translate between two systems: a position and a scalar:

- Position - refers to the position of the knob on the slider where 0.0 is the far left and 1.0 is the far right.
- Scalar - amplitude scale factor (linear scale, not in dB)

The two required functions must be the inverse of each other otherwise unusual slider behaviour will result.

The other functions and constants in the definition section are for convenience only and can be referenced by the two required functions.

Not that literal numeric values can be explicitly stated (eg: the slope property above), but expressions must be quoted (eg: the zeroDbPos above)

For details on writing expressions and the set of available functions, see the [Expression Reference](#).

## "uiKinds" Section

The uiKinds section includes additional information for different elements in Cantabile's user interface and is further broken down into three sections:

- `horizontalSlider` - the smaller always visible version of the horizontal slider
- `horizontalSliderPopup` - the popup temporary view of the slider when clicked on
- `levelMeter` - level meter indicators

Within each section is a set of "ticks", each with the following properties:

- `kind` - major, minor, hotStart, hotEnd or clipIndicator (see below)
- `scalar` - the position of the tick in scalar units (can be an expression referencing anything in the definitions section)
- `label` - the label displayed on the tick mark
- `snap` - whether to snap to this tick when using the slider in snapping mode (Ctrl key pressed). Defaults to true.

The kind setting determines the type of tick mark:

- `major` - a major tick mark
- `minor` - a minor tick mark (displayed smaller)
- `hotStart` - the position on a level meter at which the level meter indicator starts to turn orange
- `hotEnd` - the position on a level meter at which the level meter is fully orange
- `clipIndicator` - the position on a level meter at which the clip indicating border lights up

Here is an example:

```
"uiKinds":
{
  "horizontalSliderPopup":
  {
    "ticks":
    [
      { "scalar": 0, "label": "-\u221E", "kind": "major" },
      { "scalar": "fromDb(-45)", "label": "-45", "kind": "minor" },
      { "scalar": "fromDb(-30)", "label": "-30", "kind": "minor" },
      { "scalar": "fromDb(-20)", "label": "-20", "kind": "minor" },
      { "scalar": "fromDb(-15)", "label": "-15", "kind": "minor" },
      { "scalar": "fromDb(-9)", "label": "-9", "kind": "minor" },
      { "scalar": "fromDb(-6)", "label": "-6", "kind": "minor" },
      { "scalar": "fromDb(-3)", "label": "-3", "kind": "minor" },
      { "scalar": 1, "label": "0", "kind": "major" },
      { "scalar": "fromDb(3)", "label": "+3", "kind": "minor" },
      { "scalar": "fromDb(6)", "label": "+6", "kind": "minor" },
      { "scalar": "fromDb(9)", "label": "+9", "kind": "minor" },
      { "scalar": "fromDb(12)", "label": "+12", "kind": "minor" },
      { "scalar": "fromDb(15)", "label": "+15", "kind": "minor" },
      { "scalar": "fromDb(18)", "label": "+18", "kind": "minor" },
    ],
  },
  "horizontalSlider":
  {
    "ticks":
    [
      { "scalar": 0, "label": "-\u221E", "kind": "major" },
      { "scalar": "fromDb(-30)", "label": "-30", "kind": "minor" },
      { "scalar": "fromDb(-15)", "label": "-15", "kind": "minor" },
      { "scalar": "fromDb(-6)", "label": "-6", "kind": "minor" },
      { "scalar": 1, "label": "0", "kind": "major" },
      { "scalar": "fromDb(6)", "label": "+6", "kind": "minor" },
      { "scalar": "fromDb(15)", "label": "+15", "kind": "minor" },
    ],
  },
  "levelMeter":
  {
    "ticks":
    [
      // These get displayed on the level meter as ticks
      { "scalar": "fromDb(-30)", "kind": "minor" },
    ],
  },
}
```

```

    { "scalar": "fromDb(-15)", "kind": "minor" },
    { "scalar": "fromDb(-6)", "kind": "minor" },
    { "scalar": 1, "kind": "major" },

    // These describe how the transition to hot orange works
    { "scalar": "fromDb(-6)", "kind": "hotStart" },
    { "scalar": 1, "kind": "hotEnd" },

    // Hitting this point turns on the border clip indicator
    { "scalar": 1, "kind": "clipIndicator" },
  ]
}

```

## Language Translations

Cantabile supports various language translations. For more information on the available language translations, see here:

- <https://bitbucket.org/toptensoftware/cantabiletranslations>

## Multiple Configurations

Cantabile supports multiple sets of global settings. You might for example have one set of settings for your home studio and another for live performance with a band. Each configuration might have a different audio driver, different global MIDI filters, different audio port mapping etc...

The configuration to be used is specified with the command line option `/config:<nameofconfig>` and usually these would be specified in the short cut used to start Cantabile.

To create a shortcut for a custom configuration:

1. Right click on the Windows Desktop and choose New → Shortcut
2. Click the Browse button and locate the appropriate version of Cantabile.exe in one of the following locations:
  - C:\Program Files\Topten Software\Cantabile 3\Cantabile.exe
  - C:\Program Files (x86)\Topten Software\Cantabile 3\Cantabile.exe
  - C:\Program Files\Topten Software\Cantabile 4\Cantabile.exe
  - C:\Program Files (x86)\Topten Software\Cantabile 4\Cantabile.exe
3. After locating Cantabile.exe, in the text box add `/config:"somenamename"` like this:

×

← 📄 Create Shortcut

### What item would you like to create a shortcut for?

This wizard helps you to create shortcuts to local or network programs, files, folders, computers, or Internet addresses.

Type the location of the item:

Browse...

Click Next to continue.

Next Cancel

4. Click Next and give the shortcut a name:

×

← 📄 Create Shortcut

### What would you like to name the shortcut?

Type a name for this shortcut:

I

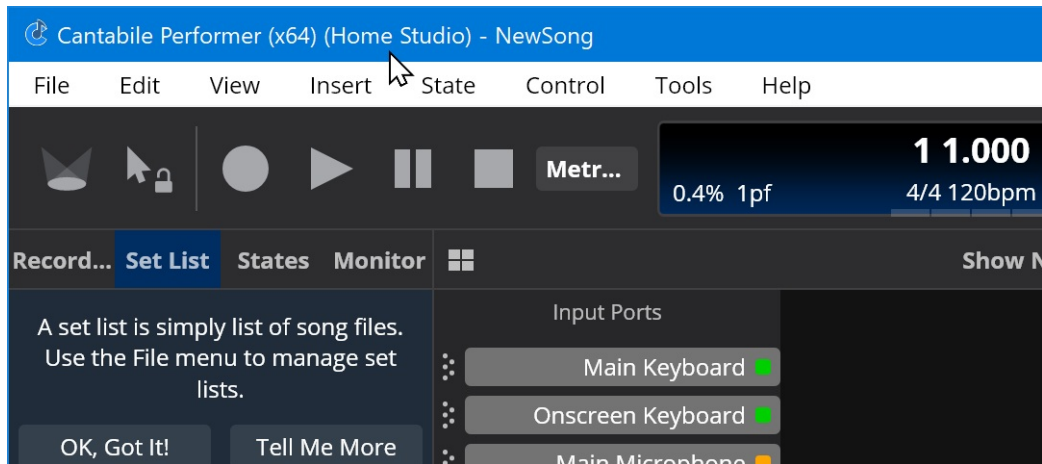
Click Finish to create the shortcut.

Finish Cancel

5. Click Finish and you should have a new shortcut on your desktop.

6. Repeat for as many different configurations as you need.

When running with a custom configuration, its name is displayed in the title bar:



## Sharing Songs Files between Configurations

For a song to work correctly in multiple configurations you just need to make sure that the set of audio and MIDI ports the song uses is available in each configuration.

For example if you have a MIDI input port named "Main Keyboard", so long as all configurations have a MIDI port with that name the song will connect to that port. Of course there's no need for "Main Keyboard" to be mapped to the same physical device in each configuration.

## Cantabile Rebranding Kit

Cantabile Lite can be rebranded as a standalone plugin host using the Cantabile Rebranding Kit (CRK). This kit is supplied on a case by case basis to plugin vendors and this page describes the process for creating a rebranded version of Cantabile.

Rebranding Cantabile allows for:

- Changing the application name (to a name approved by Topten Software)
- Changing the company name of the folder where user settings are stored
- Loading of plugins from a pre-determined path independent of the user's plugin path settings
- Custom application icon
- Custom song file extension
- Custom welcome screen graphics (ie: first run wizard)
- Custom website links in the About page
- Custom website links in the Help menu
- Custom default settings (eg: screen layout, window size etc...)
- Display of plugin version information in Cantabile's about page
- Inclusion of a default song used for new song files (that would typically have the vendors plugin preloaded).
- Removes the need for Cantabile license registration
- Feature set equivalent to Cantabile Lite

## Prerequisites

In order to produce a rebranded version of Cantabile you will need the following which will be supplied by Topten Software when use of the CRK is approved.

- A copy of the crksigntool (also requires a valid .NET Framework 4.6 installation)
- A signing key for your company (a \*.signkey.json file)

Without these files a rebranded version of Cantabile can't be produced.

## The branding.json File

On startup Cantabile looks for a file named branding.json in the same directory as the Cantabile executable file. If found, the file is loaded, its embedded cryptographic hash signature is verified and if valid the settings in the file are used to run Cantabile in rebranded mode. If the file isn't found, or the cryptographic hash isn't valid Cantabile runs as the regular non-rebranded version of Cantabile.

The contents of branding.json are explained by comments in the sample file (see below).

Any external files referenced from branding.json are assumed to be relative paths to the branding.json file.

## HiDPI Graphics

For any .png file referenced in branding.json, Cantabile will also look for hidpi versions with @2x, @3x, @4x etc... suffix to the base file name.

Eg: suppose you have a 64x64 icon file named Icon.png. On high DPI monitors Cantabile will also look for a hi-dpi version named Icon@2x.png which should be 128x128 pixels.

## Signing the branding.json file

Before Cantabile will use branding.json it must be signed with a cryptographic hash using the crksigntool.exe command line utility. For example:

This will sign the file branding\_unsigned.json and write the output to branding.json.

```
crksigntool branding_unsigned.json --out:branding.json
```

You can also sign a file inplace, but note that all comments will be removed:

```
crksigntool branding.json
```

In order to sign the file the tool must be able to locate your \*.signkey.json file (as supplied by Topten Software). By default the tool looks for a file with that extension first in the current directory and then in the same directory as the crksigntool.exe file - so usually you shouldn't need to explicitly specify it. But if you do, use the --key: command line argument:

```
crksigntool branding.json --key:/some/other/path/myfile.signkey.json
```

## Configuring a Default Song

The branding.json file can include a reference to a default song. This song will be loaded whenever the user chooses File|New or anytime a new empty song file is required (eg: first run).

By creating a song file with your plugin loaded, Cantabile will launch with your plugin loaded and ready to go.

Typically you'll also want the plugin's UI to be shown as the primary user interface in Cantabile's main window. To do this:

1. Start Cantabile (either rebranded or regular Cantabile Lite)
2. Load your plugin and configure its input and output routes (if required)
3. Double click the plugin to bring up it's user interface
4. From the hamburger menu at the top right, choose "Docked" - this will embed the plugin UI into Cantabile's main window
5. From the same hamburger menu, choose "Primary View" - this tells Cantabile to switch to that plugin view tab when the song is loaded.
6. Save the song file

Place the saved song file in the same directory as the branding.json file and reference it from the "defaultSong" setting. If you created the file using the non-rebranded version of Cantabile, you should also change its file extension to match your custom song file extension.

## Testing

To test your rebranded version of Cantabile, download and extract the Cantabile.zip package from the [Cantabile release notes page](#). Rebranding was introduced in build 4053 so you'll need that version or later.

Next, copy your signed branding.json file, along with any referenced asset files to the Cantabile installation directory. Run Cantabile by running Cantabile.exe in the extracted directory.

Remember after making changes to the branding.json file you'll need to resign the file again in order for Cantabile to load it.

## Redistribution and Installation

Approval to use the Cantabile Rebranding Kit also includes permission to redistribute Cantabile either via download or on physical media.

Your installation program should:

- Copy all the files in the above mentioned zip package to a directory under your product's installation folder.
- Copy your signed branding.json file and any referenced asset files in the same directory.
- Make sure Microsoft .NET 5.0.8 (or later) is installed.
- Run cantabile.exe /regserver to register file associations.
- Create a shortcut to Cantabile.exe that the user can use to launch the application.

You can choose to include both the x86 and x64 versions, or just one or the other. Note that there are separate .NET installations for each platform, so if you include both platforms, you'll also need to install both versions of the .NET runtime.

See [Installation](#) for details on required runtime versions.

Further information is also available in the readme file included in the .zip package. Note however the section on configuring the network isn't required for CRK installations since the network server isn't supported in Cantabile Lite.

## Sample branding.json File

The following is a sample branding.json file for a fictional "Rocket" plugin by company "Acme".

```
{
  // Overrides the icon file on the main window and used for file extension associations
  "iconFile": "Rocket.ico",

  // Overrides the icon displayed in the about page
  // Expected size: 64x64 pts
  // HiDPI versions Rocket@2x.png etc.. will be use automatically if present
  "iconImage": "Rocket.png",

  // Path to plugin folder
  // This path will always be included before the user-paths configured in options
  // Omit either path if not distributing for that platform.
  "pluginPath64": "..\\Plugin64\\",
  "pluginPath32": "..\\Plugin32\\",

  // The base prog id for file association entries in the system registry
  "baseProgId": "Rocket3",

  // Base string for file extensions
  // eg: "Rocket" => "*.RocketSong"
  "baseExtension": "Rocket",

  // A title string to be displayed with the version number extracted from the file below
  "versionTitle": "Rocket Plugin",

  // Path to a file from which to extract version information
  "versionFile": "..\\Plugin64\\Rocket.dll",

  // A default song to load whenever File → New is selected.
  // Typically this would be configured with the plugin already loaded.
  "defaultSong": "defaultSong.RocketSong",

  // Overrides the banner image shown on the welcome/getting started pages
  // NB: two versions of this depending on whether dark or light theme is selected
  // Expected size: 420x80 pts and (@2x hidpi versions used if present)
  "welcomeGraphic_dark": "WelcomeGraphic_Dark.png",
  "welcomeGraphic_light": "WelcomeGraphic_Light.png",

  // Links to be displayed in the about page
  "aboutLinksTitle": "Acme",
  "aboutLinks": [
    {
      "title": "Website",
      "target": "https://acme.com"
    }
  ],

  // Links to be displayed in the help menu
  "helpLinksTitle": "Rocket",
  "helpLinks": [
    {
      "title": "FAQ",
      "target": "https://acme.com/faq"
    },
    {
      "title": "Support",
      "target": "https://acme.com/support"
    },
    {
      "title": "Updates",

```



```

        "target": "https://acme.com/updates"
    }
},

// Default setting overrides. These settings will be used when there's no
// settings file found during startup. All settings can be overridden by
// the user after first run. For other global settings refer to a complete
// 'settings.json' file (see Cantabile → Tools → Open Settings Folder)
"settings": {
    "mainWindow": {
        {
            // Default to hidpi mode with plugin up-scaling enabled
            "dpiMode": "Window",

            "normalViewState": {
                {
                    // Adjust to suit the default size of your plugin
                    "placement": "auto,auto,925,725,False:False,1",

                    // Hide the side panel
                    "sidePanelVisible": false
                },

                // Hide all panel tips
                "hiddenTips": 67
            }
        },
    }
},
}

```

## Resource Folder

When Cantabile is attempting to locate various resource files (eg: [theme](#) files and [custom gain control curves](#)), it looks first in the resource folder and then in the same location as the Cantabile executable file.

The resource folder can be configured in Options → File Locations.

After changing the resource folder, the application must be restarted before it takes effect.

## SysEx Macros

When entering MIDI SysEx data Cantabile supports the following macros.

*Note: Unlike version 2, version 3 doesn't support custom user-defined macros. This may be supported in a future version. If this is something you require, please [get in touch](#)*

hibyte(x)	- gives the high byte of a word eg: hibyte(0x1234) gives 0x12
lobyte(x)	- gives the low byte of a word eg: lobyte(0x1234) gives 0x34
ascii(x)	- generate ascii data for string x eg: ascii("apple") gives 0x61 0x70 0x70 0x6c 0x65
strlen(x)	- return the length of string x eg: strlen("apple") gives 5
strcat(x,y)	- join strings x and y
add(x,y)	- add x and y
sub(x,y)	- subtract x and y
mul(x,y)	- multiply x by y
div(x,y)	- divide x by y
mod(x,y)	- remainder of x divided by y
and(x,y)	- x and y (bitwise)
or(x,y)	- x or y (bitwise)
xor(x,y)	- x xor y (bitwise)
shl(x,y)	- shift x left by y bits
shr(x,y)	- shift x right by y bits
hex(x)	- convert number x to hex (and add 0x prefix)
dec(x)	- convert number x to decimal
bytelen(bytes)	- calculate the length of a byte stream
byteswap(bytes)	- byte swap the byte stream
midivarlen(x)	- midi encode variable length number x
byte(x)	- Format x as a hex byte
word(x)	- Format x as a hex word
dword(x)	- Format x as a hex dword
qword(x)	- Format x as a hex qword
repeat(x,y,d)	- Repeat x, y times with optional delimiter d

## Variables

Cantabile supports the following string expansion variables. Normally the availability of these is indicated by a pop out

menu next to the field that supports them.

Note that the body of an expansion variable is not limited to a simple variable name and supports a expression evaluation.

See also the [Function Reference](#).

These variables are also available in [Sys-ex Expressions](#).

## Supported Variables

```
$(am)
$(AppData)
$(AudioDriver)
$(BufferSize)
$(BatteryLevel)
$(BatteryState)
$(BatteryTime)
$(CommonAppData)
$(CommonDesktop)
$(d)
$(day)
$(dd)
$(Desktop)
$(dy)
$(Favorites)
$(FirstEffectPluginName)
$(FirstEffectPluginProgramName)
$(FirstMediaFile)
$(FirstMediaFileFolder)
$(FirstMediaFileTitle)
$(FirstMediaFileDisplayName)
$(FirstPluginName)
$(FirstPluginProgramName)
$(FirstSynthPluginName)
$(FirstSynthPluginProgramName)
$(h)
$(h24)
$(hh)
$(hh24)
$(IsAutoRecord)
$(IsMainsPowered)
$(IsBatteryCharging)
$(IsRecording)
$(Load)
$(LoopCount)
$(LoopIteration)
$(LoopInfo)
$(LoopMode)
$(MasterMediaFile)
$(MasterMediaFileFolder)
$(MasterMediaFileTitle)
$(MasterMediaFileDisplayName)
$(m)
$(MemoryPageFaults)
$(MemoryUsage)
$(MemoryWorkingSet)
$(min)
$(mm)
$(mmm)
$(mmmm)
$(MyDocuments)
$(MyMusic)
$(MyPictures)
$(NextState)
$(NextSong)
$(PreviousState)
$(PreviousSong)
$(ProgramFiles)
$(ProgramFilesX86)
$(SampleRate)
$(sec)
$(SelectedMediaFile)
$(SelectedMediaFileFolder)
$(SelectedMediaFileTitle)
$(SelectedMediaFileDisplayName)
$(SelectedPluginName)
$(SelectedPluginProgramName)
$(SongFile)
$(SongFolder)
```

```

$(State)
$(StateProgram)
$(SongTitle)
$(SetListFile)
$(SetListFolder)
$(SetListTitle)
$(Song)
$(SongNumber)
$(SongPart) or $(StateIndex)
$(SongPartCount) or $(StateCount)
$(SongProgram)
$(System)
$(SystemX86)
$(Temp)
$(Tempo)
$(TimeSignature)
$(TimeSignatureDenominator)
$(TimeSignatureNumerator)
$(TransportPosition)
$(TransportPositionMusical)
$(TransportPositionRealtime)
$(TransportPositionShort)
$(TransportState)
$(w)
$(Windows)
$(wm)
$(ww)
$(wmm)
$(yy)
$(yyyy)

```

## MIDI Controller Variables

The following variables reflect the current MIDI state of the on-screen keyboard device.

By default, these variables, return the current value for MIDI channel 1. To access other channels, include the channel number as an additional parameter eg: `$(cc(23,5))` to get the value of CC #23 on channel 5.

```

$(cc(controllerNumber))
$(finecc(controllerNumber))
$(program())
$(bankedProgram())
$(pitchBend())
$(channelPressure())
$(rpnCoarse(paramNumber))
$(rpnFine(paramNumber))
$(nrpnCoarse(paramNumber))
$(nrpnFine(paramNumber))

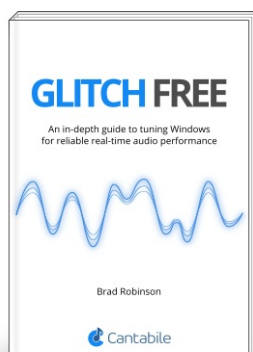
```

See also the [Function reference](#) for other functions.

## Tuning for Reliable Glitch Free Performance

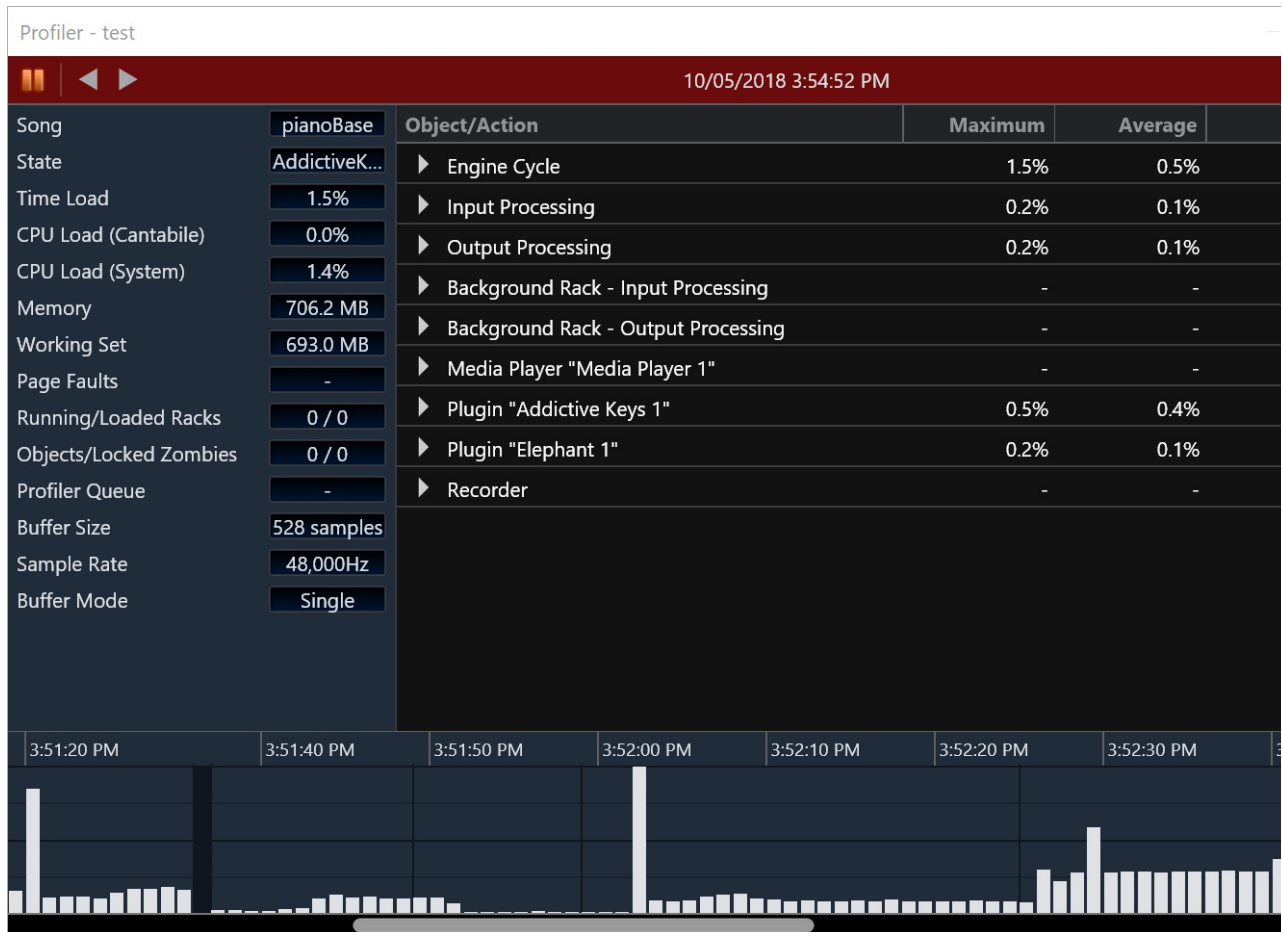
Before using any PC for live performance work the machine should be configured for reliable, glitch free performance. This is beyond the scope of this guide however we've written an entire book on the subject!

Get your free copy of our eBook "Glitch free" here - <https://www.cantabilesoftware.com/glitchfree>



# Performance Profiler

Cantabile includes a comprehensive Performer Profiler to provide deep insight into which parts of the audio processing pipeline and/or plugins might be causing load and performance issues:



## Accessing the Profiler

The Profiler is available from the View menu... View → Profiler.

## Profiler History

The profiler records a history of audio engine processing for the previous five minutes. The graph at the bottom of the profiler displays a histogram bar for each one second interval over the recorded period.

To see a detailed breakdown of a particular one second interval:

1. Click on that bar in the graph.
2. The real-time updating of the profiler will pause and the side panel and detail view will show what happened during that period.
3. Switch to other intervals to investigate other possible issues.
4. Press the Pause button (top left of Profiler window) to return to real time profiling data.

You can also navigate through the profiler history using the Left and Right arrow keys.

## Other Graph Indicators

The profiler graph also shows:

- a fine vertical line where the currently loaded song or song state changed.
- a thick vertical black bar where the audio engine was stopped or profiler was disabled.
- red bars for any interval where the time load exceeded 100%

## Side Panel Metrics

The side panel displays metrics for the currently selected interval. These metrics essentially match those displayed in the

main [Monitor Panel](#).

## Detail Panel

The main content area of the profiler displays a break down of each operation that the audio engine performed during that one second interval. The displayed percentages are calculated as a time percentage of the audio buffer length (see [Understanding Performance Metrics](#) for a more detailed explanation of this).

Each top level row in the detail panel represents a single audio engine "Work Item". Each work item is a task that was scheduled to execute on one of the audio engine's worker threads. Since these items can execute in parallel the total percentages of these items might exceed the time load percentage of the audio cycle as a whole - this is because these work items can execute in parallel.

Note too that since the displayed interval is an aggregate of all audio cycles processed during a one second interval sometimes you'll see plugins from two different songs in the same interval. This means that at the start of the interval one song was processing while at the end of interval the other song had loaded and started processing.

When the profiler detail level is set to anything higher than Normal you can expand each row in the detail view to get a breakdown of other operations. Any sub-items of the root items are displayed in execution order. Even so, sometimes the displayed rows might be somewhat unintuitive (or duplicated entries). This is a side effect of the profiler coalescing multiple audio processing cycles in to a single one second interval.

## Exporting Profiler History

Sometimes it might be necessary to send a profiler log to Topten Software for analysis. To do this, from the menu button at the top right corner of the profiler window, choose "Save" to export the history.

You can also re-import a previously saved profiler history using the Open command. To close a previously opened history file and return to real-time profiler data, just click the Pause button.

## Profiler Settings

The profiler has a couple of settings you can adjust via the Options → Diagnostics page:

- Detail level - adjust this to provide a more detailed breakdown of profiler timing
- History Length - how much history to keep (5 minutes to 2 hours).

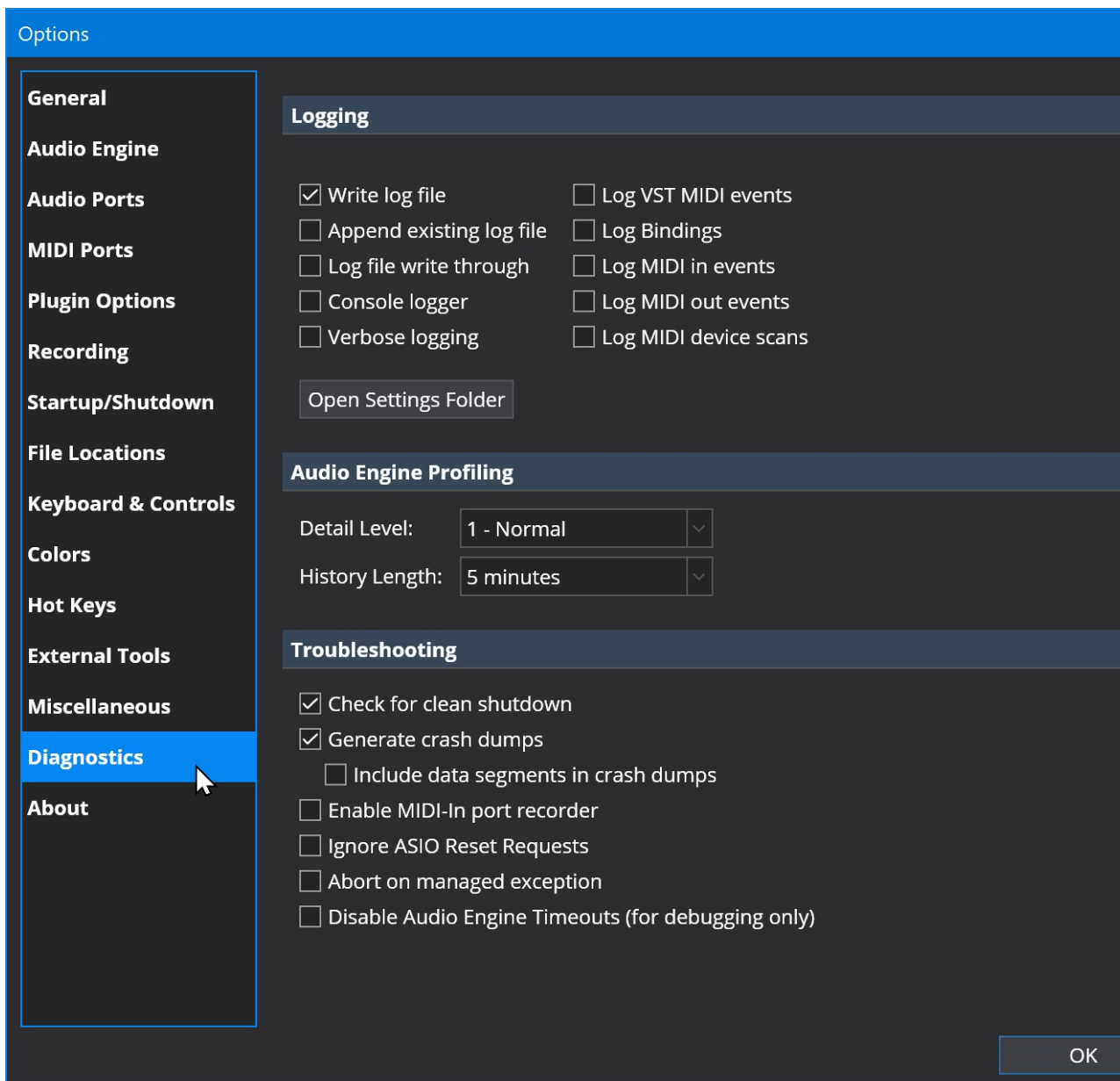
Usually the "Normal" detail level is enough however if you're trying to diagnose a specific issue, turning up the detail level can provide additional insight (at the possible expense of additional processing load).

Note that profiler history is kept in memory - the longer the history the more memory will be used. Be careful setting this to a large value when the detail level is also turned up.

## Diagnostic Options

Cantabile supports various settings for diagnosing issues.

Many of these options when enabled will affect Cantabile's performance and should be turned off once you've finished diagnosing any issues. Cantabile will display a warning on startup if one or more of these options might adversely affect performance.



#### Write Log File

Writes a log file listing out operations as they occur in internally to Cantabile. The log is written to a file named log.txt in the settings folder.

#### Append Existing Log File

When not selected, the previous log file is overwritten on each run. (A copy of one previous log file is kept as log-previous.txt).

#### Log File Write Through

Disables disk write caching on the log file to ensure every log entry is written. Prevents losing some log messages immediately prior to a crash or hang. Can severely affect performance - don't use unless necessary.

#### Console Logger

Displays the log messages in a separate popup window while Cantabile is running. Useful for diagnosing events in real-time.

#### Verbose Logging

Logs more stuff :)

#### Log VST MIDI Events

Logs all MIDI events sent to plugins

#### Log MIDI In Events

Logs all incoming MIDI events

#### Log MIDI Out Events

Logs all outgoing MIDI events

#### Open Settings Folder

Launches Windows Explorer showing Cantabile's settings folder.

#### Generate Crash Dumps

Captures detailed error information if Cantabile a loaded plugin crashes. These files are essential in helping us resolve issues with Cantabile and after a crash dump is generated you'll be prompted to send it for diagnosis. Please do.

#### Include Data Segments in Crash Dumps

Captures a lot more information in the crash dump. So much that the file will be typically too large to email. Sometimes we'll ask for this to be turned on and we'll provide an upload location to send the file.

#### Abort on Managed Exception

Some exceptions can be safely ignored. Although these issued need to be investigate Cantabile will typically continue running to allow you to save your work. Sometimes we'll ask to turn this option on before reproducing an issue so we can capture additional information about the problem.

#### Disable SSE and SSE2 Optimizations

Disables various floating point math optimizations. You should never need to use these. *Removed in build 3500 and later*

#### Disable Audio Engine Timeouts

Cantabile tries to detect if the audio engine becomes unresponsive. This option disables that. You should never need to use this unless otherwise instructed.

#### Don't Check For Clean Shutdown

If at startup Cantabile detects that it didn't shutdown cleanly on the last run you'll be prompted with some recovery options. This option disables the recovery screen.

#### Ignore ASIO Reset Requests

Work around for some problematic ASIO drivers that send excessive/spurious reset requests. We'll generally advice to turn this on if we see appropriate error messages in debug logs you send.

## Settings Folder

Cantabile stores all its global settings in the settings folder. The exact location of this file will depend on the version of Windows you're running, whether you're running the x86 or x64 version of Cantabile and whether you're running with an [alternate configuration](#).

### Locating the Settings Folder

The easiest way to locate the settings folder it to:

1. Start Cantabile
2. From the Tools menu, select Options
3. Go the the Diagnostic page
4. Click the "Open settings and log file folder"

This will launch Windows Explorer in the correct folder.

### Moving the Settings Folder

Sometimes you might want to move the settings folder to a different location. This can be handy if you want to backup or synchronize your settings across machines with tools like Dropbox.

You can change the location of the settings folder by editing the config.json file that can be found in the same folder as Cantabile.exe:

```
{
  "settingsFolder": "%USERPROFILE%\\Documents\\Cantabile\\Settings"
}
```

Alternatively, starting with Cantabile 4, you can also set the settings folder with a command line option:

```
Cantabile.exe --settingsFolder:"%USERPROFILE%\\Documents\\Cantabile\\Settings"
```

The location specified here is the base settings folder location and will be appended with "Cantabile 4.0 (x64)" and the config name as per normal.

Note that this setting applies to all user accounts which is why in the above example it includes the %USERPROFILE% environment variable so that each user's settings are kept separate.

Once you've changed this setting don't forget to move all the files from the old location so you don't need to reconfigure everything.

Finally, remember that the settings folder stores settings relevant to your hardware configuration. If you're using this synchronize settings across machines you'll need to be running the same hardware on each machine (or you'll need to use different configuration names on each machine).

## Setting Files

#### settings.json

This file is the main file in which all of Cantabile's global settings are saved.

#### plugins.json

Information all plugins shown by Cantabile's Insert Plugin window.

#### plugins.cache.json

Cached information from the previous plugin scan. Used when performing a quick plugin scan to save re-scanning previously scanned plugins.

#### plugins.user.json

User settings for plugins such as categories.

log.txt

Cantabile's debug log used for diagnostic purposes.

Cantabile-xxxxxxxxxxxxxxxxxxxxxxxxxxxx.zip

Crash reports files contained detailed diagnostic information collected if Cantabile crashes. Typically these will be sent to Tipton Software by the crash reporter program.

## Prevent Memory Paging Options

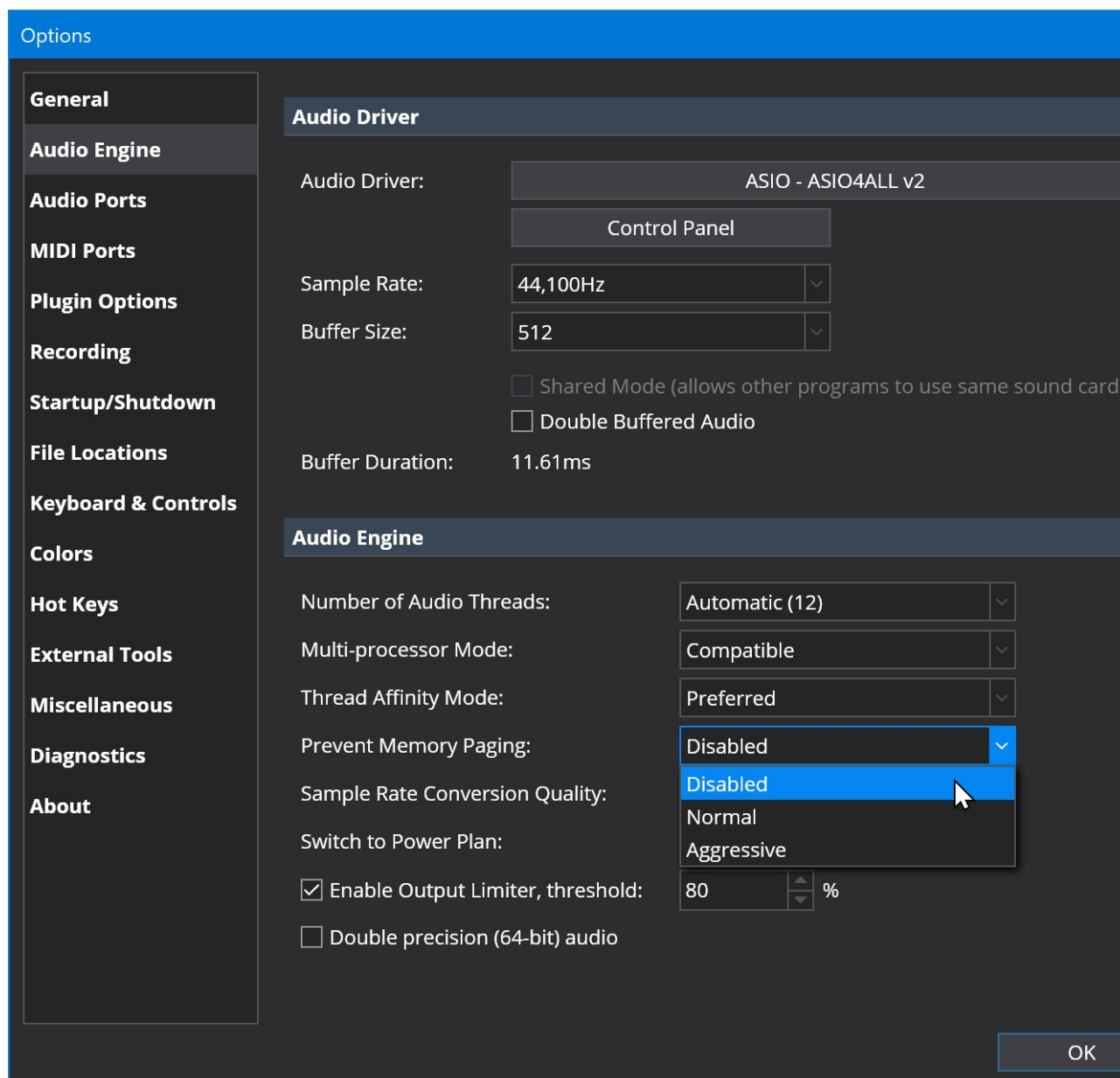
When using large sample based instruments memory paging can have an adverse effect on performance and sometimes lead to audio drop outs. This happens because Windows will sometimes unload sections of memory that it thinks are not actively being used to make room for other programs.

For example when a machine is left running but idle overnight Windows will often page out large sections of memory for no apparent reason. When returning to the machine in the morning there can be considerable audio dropouts as each page is memory page read back from disk.

Cantabile's Prevent Memory Page options helps reduce these problems by periodically "touching" each page of memory to let Windows know that it's still in use (even if it hasn't been used in a long time).

## Enabling Memory Paging Options

The options to prevent memory paging are on the Audio Engine options page in the Options dialog.





Using this option can reduce audio drop outs for memory intensive situations for situations where the PC may be left unused for long periods of time, but needs to be ready to go at a moment's notice.

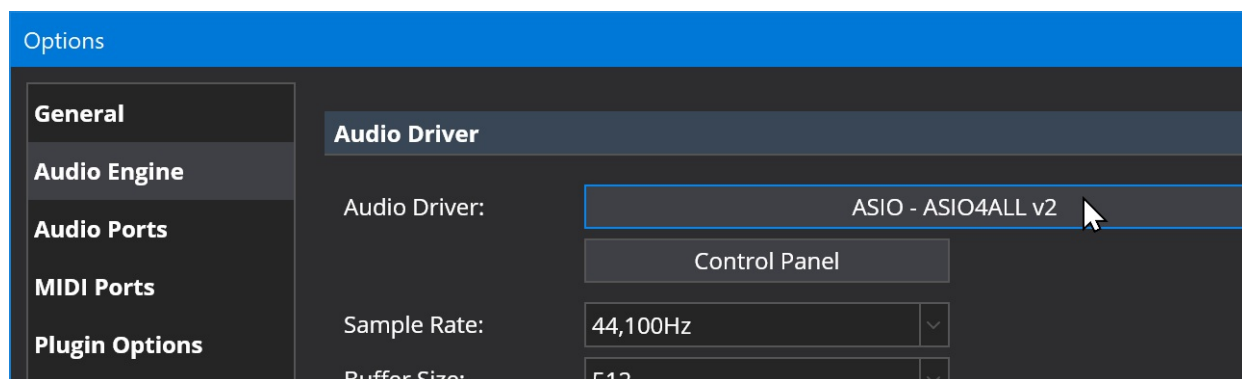
Note: only use this option if you're getting drop outs without it. By enabling this option you're circumventing Windows otherwise excellent memory management and may result in decreased performance in other applications (as well as Cantabile).

## No Sound

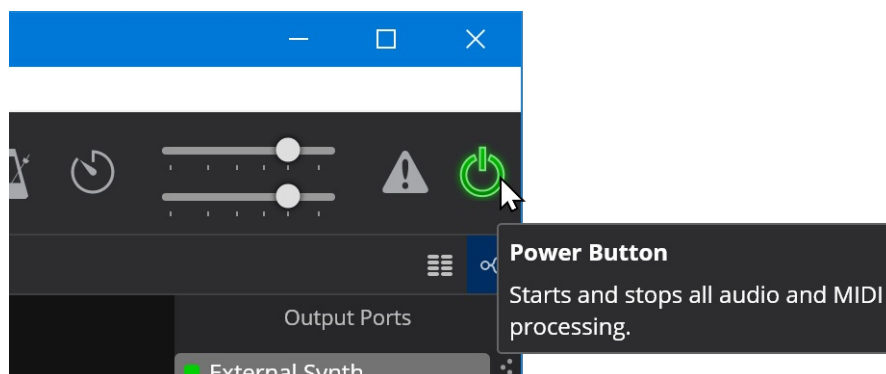
If you're having issues with no sound coming from Cantabile diagnosing the problem is often a process of elimination - eliminating possible causes until the actual problem is found.

Here's a list of things to check/try that should help get to the bottom of the issue.

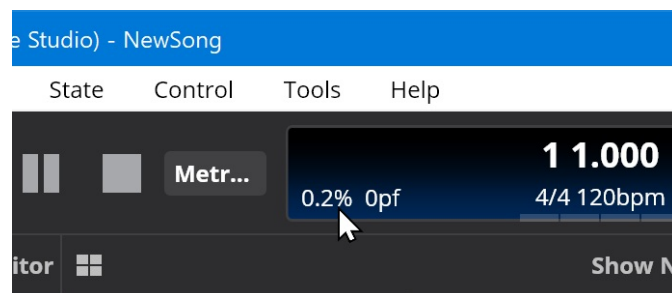
1. Make sure you have the correct audio driver selected in Options → Audio Engine. (note that the Null Audio driver will *not* create any sound)



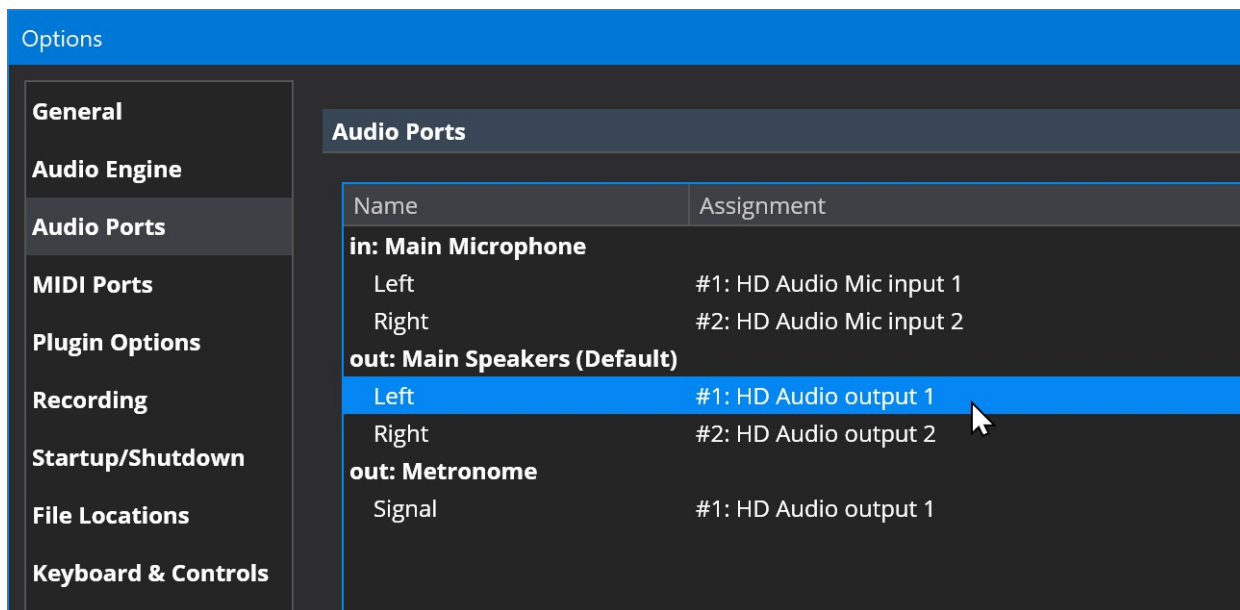
2. Make sure the audio engine is running (the power indicator at the far right of the main toolbar should be lit up green)



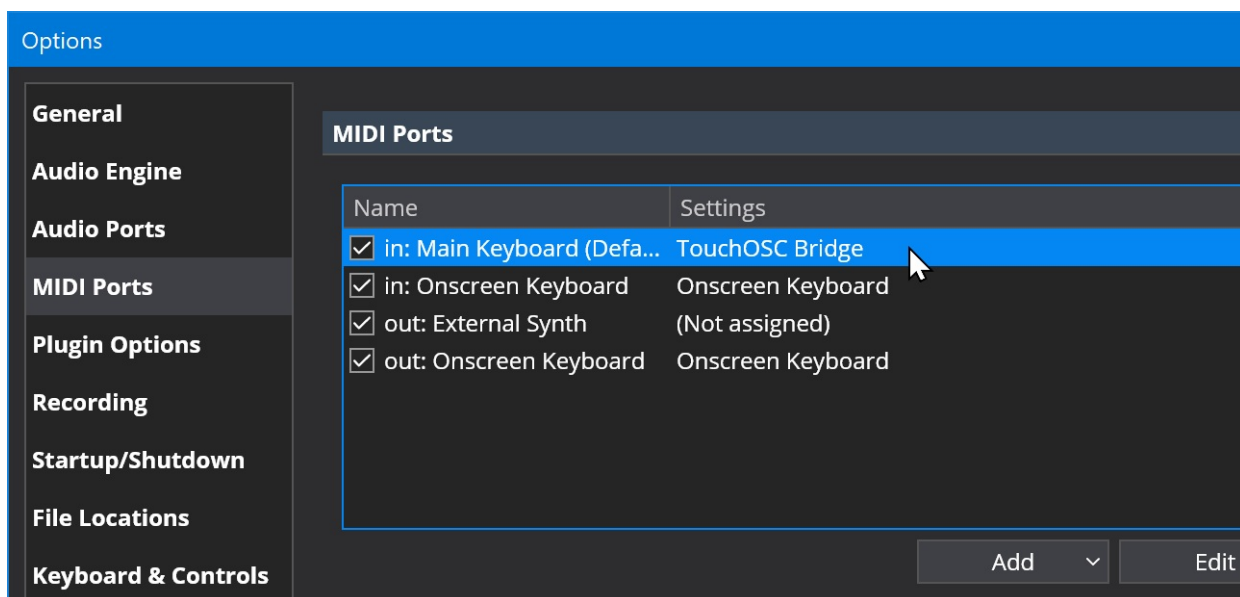
3. Make sure you don't have any other software running that's using the same audio device. If so shut those programs down and either restart Cantabile, or stop/start Cantabile's engine. Sometimes other software using hardware can be subtle - particularly for the default audio device. Don't forget to check for less obvious things like Skype, web pages in browser tabs, running virtual machines etc...
4. Check if Cantabile's load meter (in the status panel center top) is showing any load - if not the audio engine isn't running correctly, probably because the audio driver failed to start.



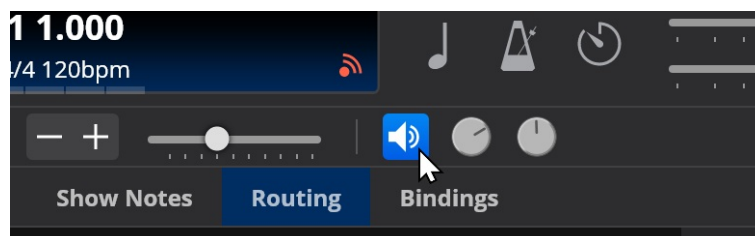
5. Check that the audio ports are correctly mapped to the audio driver in Options → Audio Ports. For a default configuration you should have Main Speakers and Metronome mapped to output channels.



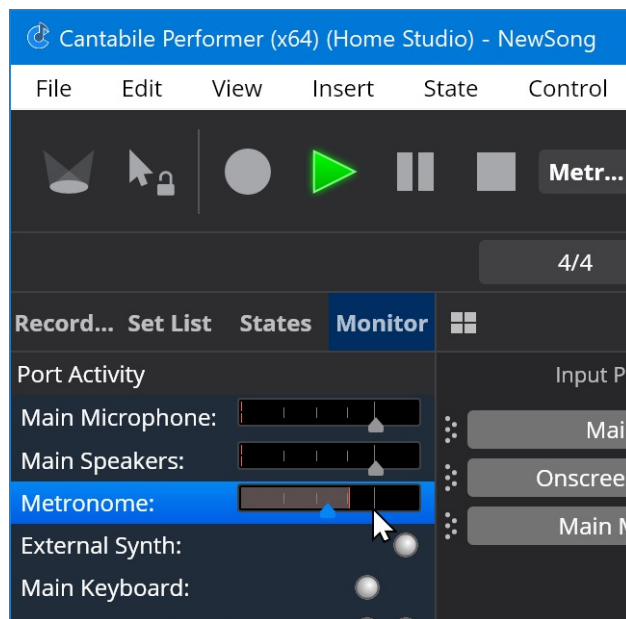
- If you're trying to play an instrument from an external keyboard, make sure the correct device is mapped to a MIDI input port. Also a handy trick is to double click on the MIDI port and also map the on-screen keyboard to the port. This will let you test using the on-screen keyboard and determine if the problem is within Cantabile or between your keyboard and Cantabile.



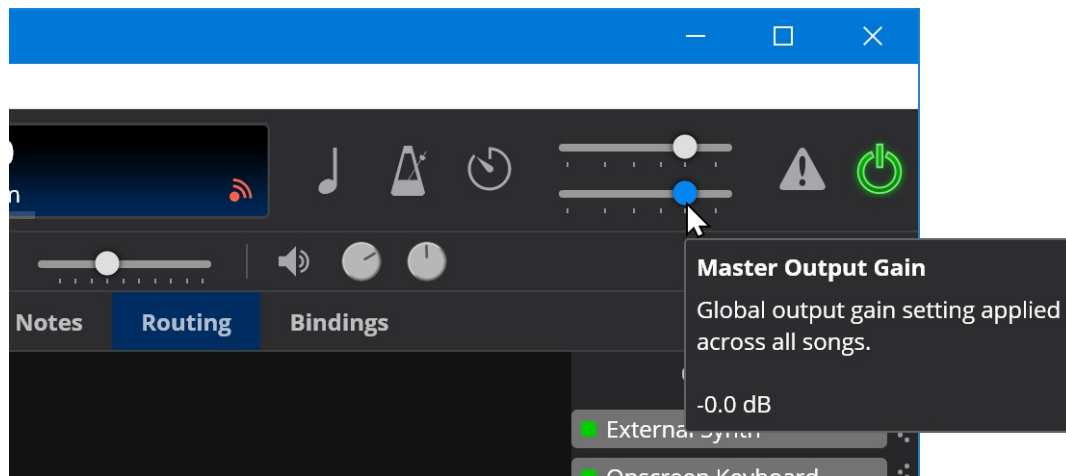
- Click the Metronome button to show the metronome bar and then turn on the metronome sounds (also check the metronome sound gain knob):



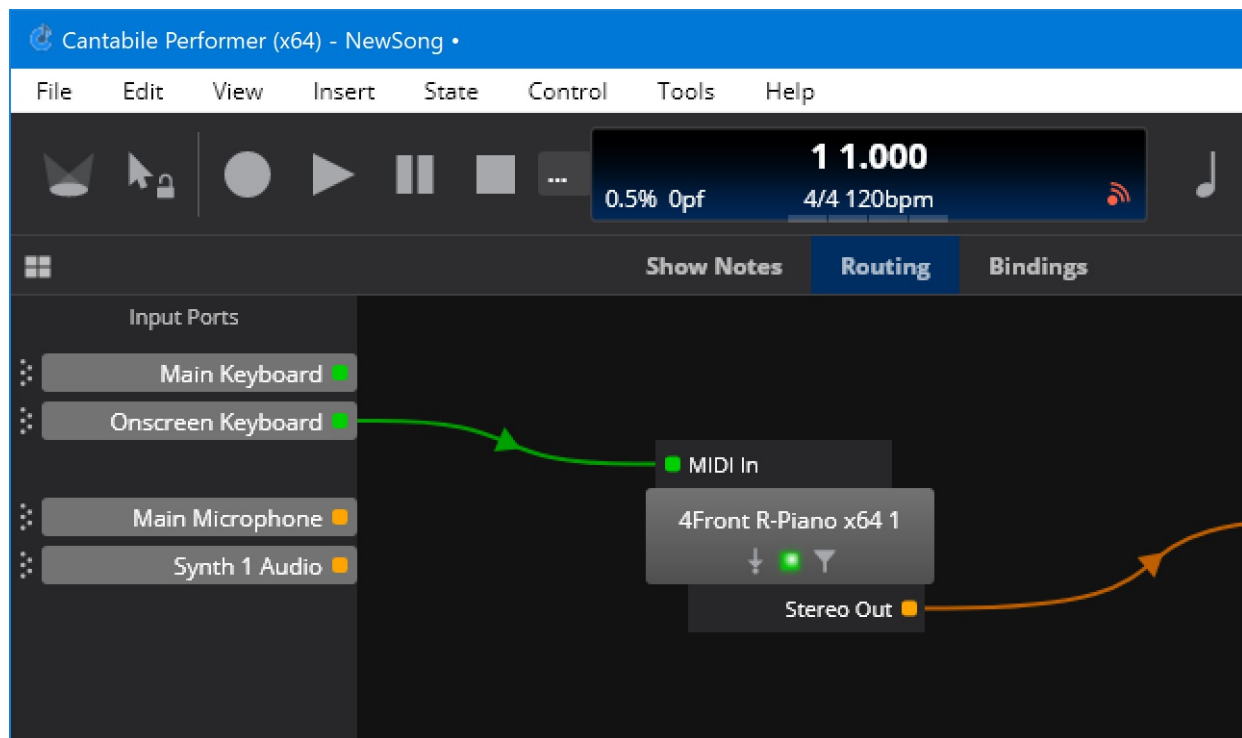
- Press the main Play button and check if you can hear the metronome sounds. If not there's something wrong with your output channel mapping, audio driver connections or sounds card to speaker connections. If the edition you're running support the monitor panel you could check too that the metronome port level meter is lighting up (as shown)



9. Make sure the main output gain settings is turned up:



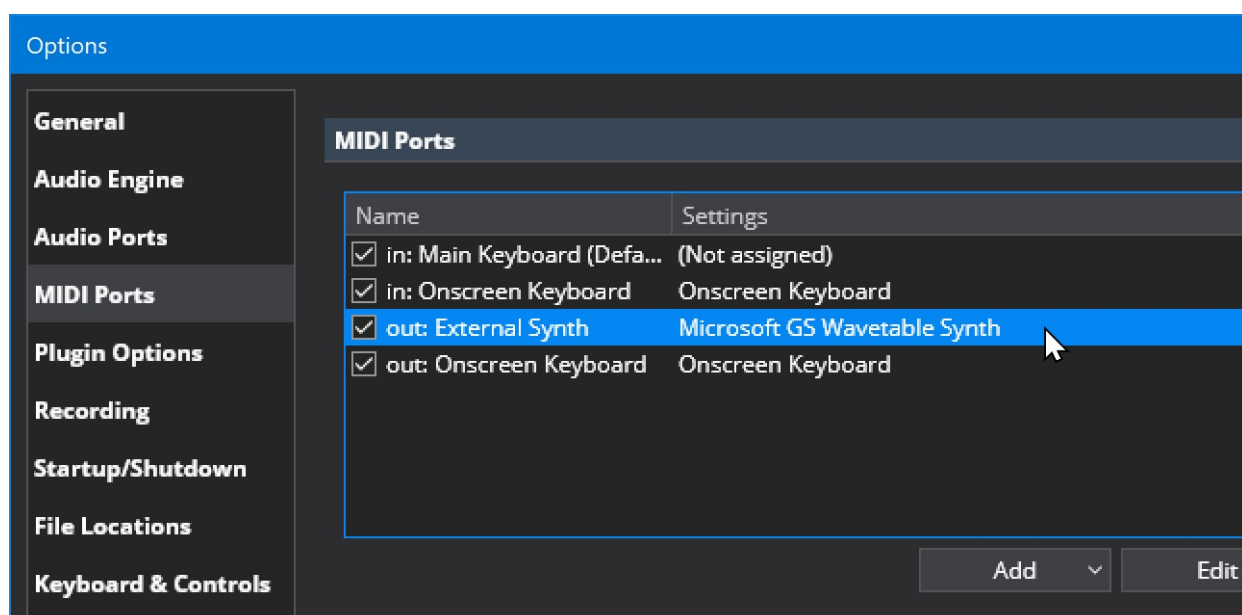
10. Try loading a VST instrument, create a MIDI route from the onscreen keyboard to the plugin and audio route from the plugin to the Main Speakers. Play some notes and check for the output level meters lighting up.



Note: 1. MIDI route from Onscreen Keyboard to mda Piano 2. Audio route from mda Piano to Main Speakers 3. Green MIDI activity indicators lit up on MIDI route 4. Audio level meter activity lit up on Audio route

11. Try switching to a different audio driver and/or device to isolate if this is a problem with one particular audio driver.
12. Try using the generic [ASIO4ALL audio driver](#). You might need to use it's advanced mode settings to enable channels for the device you want to use.
13. Try turning the audio engine off and on again. If Cantabile displays any messages during startup, sometimes the system sounds that play when showing the message can cause an audio driver not to start correctly. This often happens when using ASIO4ALL and when Cantabile's diagnostic options are turned on - Cantabile displays a performance warning about them, which plays a system sounds which causes ASIO4ALL to not start correctly.
14. Make sure you don't have the Microsoft GS Wavetable Synth MIDI device selected in Cantabile's MIDI Output device list - this device has known compatibility issues with some audio drivers.

ie: *Don't* do this:



If all of the above fails you can check Cantabile's debug log:

1. In Options → Diagnostics, turn on Console Logger
2. Restart Cantabile
3. A second window will appear with a list of log messages - look for error messages which will typically appear in red.

Finally, if you still can't get it to work please send debug report:

1. From the Tools menu, choose Open Settings Folder
2. Locate the files log.txt, log-previous.txt and settings.json
3. Close Cantabile

Send a copy of these three files and a copy of your song file to our [contact email address](#).

## Introduction to Computer Based Music

If you're new to the world of computer based music this sections explains a few of the basic concepts you'll need to understand to get the most of Cantabile. If you're already familiar with computer music you can skip this section.

### MIDI vs Audio

In computer music, there are two main ways of representing musical information:

- Audio - Audio data is a digital representation of the actual sound waves that make up sound. WAV and MP3 files are typical ways to store audio data. Computer software can be used to process audio data to apply effects such as reverb, echo etc...
- MIDI - MIDI is standard way of recording the notes that make a piece of music. MIDI doesn't necessarily define the sound of each note so a MIDI file can be played back with a different instrument to which it was recorded. MIDI is not limited to notes - it can also be used to represent other controls such as damper pedals, modulation wheel data and more.

Think of audio as similar to storing music on a CD whereas MIDI is more like sheet music.

### Synthesis, Virtual Instruments and Audio Effects

Sound synthesis is the process of taking musical notes (typically represented as MIDI data) and converting it into an digital sound. ie: the process of converting MIDI to audio.

A virtual instrument is a piece of computer software that performs this process. For example, a virtual piano accepts MIDI notes as input and generates the sound of a piano playing those notes.

An audio effect is a piece of software that manipulates an audio signal. This could be anything from making it louder or softer, to widening the stereo width or applying sophisticated reverb effects (eg: recreating the sound of a large cathedral).

### Plug-ins and Hosts

Computer music software typically falls into one of two main categories - plugins and hosts.

- A host is the main program that a user uses to create their music and provides the framework for routing audio and MIDI between external devices and loadable modules called plugins.
- A plugin is a software module that is loaded by a host to perform a specific function. Plugins fall into two main categories - instruments and effects (as described above). Other plugin types include MIDI effects (which generate MIDI), surround sound processors and analysers.

Cantabile is a host application that is designed for real-time performance. This means it's designed for connecting a MIDI keyboard, microphones and other devices and processing them in real-time.

Cantabile does not include any plugins and as such you'll need to acquire these separately. Free and commercial plugins are available from many different software developers. There are a number of different plugin formats available, though Cantabile only supports the most popular - VST. VST (short for Virtual Studio Technology) is a standard defined by Steinberg.

A good source of information on available plugins is [KVR Audio](#).

### Audio Drivers

An audio driver is a piece of software that knows how to communicate with a sound card. Cantabile uses audio drivers to allow it to work with many different types of sound cards. In order to use Cantabile you will need a compatible audio driver.

One of the most important features of a good audio driver and sound card is the ability to deliver low-latency audio. Latency is a term used to describe the time delay between the computer generating a sound and the sound being heard. It also describes the delay between receiving a sound and that sound being delivered to the computer software. Latency is measured in milliseconds and ideally you will want a latency of no more than about 10ms.

Cantabile supports two different audio driver standards:

- ASIO - short for Audio Stream Input/Output is a standard specified by Steinberg. ASIO generally offers very low latency and is the preferred type of audio driver.
- WASAPI - (Windows Audio Session API) is a Microsoft technology for audio streaming.

If your sound card came with an ASIO driver you should use that driver – particularly if it is a professional quality sound card. If your sound card did not come with an ASIO driver a good option is the excellent generic ASIO driver ASIO4ALL available from <http://www.asio4all.com>.

In addition to these standard audio driver types, Cantabile also includes a Null audio driver. The null audio driver doesn't produce any sounds but allows the audio engine to run as if a sound card was attached. This is mostly useful for using Cantabile as a MIDI only processor.

## MIDI Drivers

In addition to a good audio driver you'll also need MIDI drivers for any connected MIDI devices. Typically these come with the device and there's no choice in which driver to use. If you've lost the driver for a piece of hardware it can typically be download from the manufacturer's website.

# Understanding x64 vs x86 and 64 vs 32-bit Audio

One of the common misunderstandings about Cantabile and audio technology in general seems to be the difference between x64/x86 and 64/32-bit audio.

Cantabile comes in both x86 and x64 editions and both editions support both 32 and 64-bit audio.

## x64 vs x86

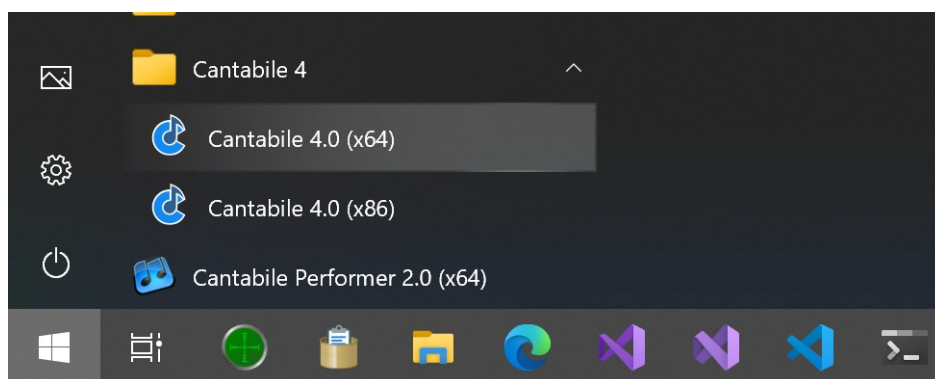
These terms refer primarily to the type of CPU and operating system on which Cantabile will run.

- x86 refers to a 32-bit CPU and operating system
- x64 refers to a 64-bit CPU and operating system

The main difference between these two platforms is the amount of RAM they can access. x86 has a physical limit of 4GB RAM (although Windows reserves the top 1GB, limiting this further to a maximum of 3GB). x64 can access more than 4GB of RAM - up to more than you'll ever need.

Cantabile comes in two editions, one for each platform. On each platform you'll need compatible operating system, drivers and plugins. ie: to use the x64 edition of Cantabile you'll need an x64 operating system (often called 64-bit Windows), audio and MIDI drivers for that operating system and x64 plugins.

To run Cantabile for a particular platform, make sure you choose the correct shortcut from the Windows Start menu:



In general if you're running a 64-bit version of Windows you should also run the 64-bit version of Cantabile. The only real exception here would be if you have plugins that aren't available as x64 and you don't need more than 3Gb of RAM.

If most of your plugins are available as x64 with only a few not available as x64, Cantabile supports a technology called jBridge that allows loading x86 plugins in the x64 edition of Cantabile and vice versa - see [jBridge](#).

## 32 vs 64-bit audio

These terms refer to the size of a single audio sample - 32 or 64 bits (also called single and double precision) and is related

to audio quality. A more precise number (64-bit) obviously allows for a more accurate representation of a sound.

That said, 32-bit audio samples are generally more than good enough for nearly all applications and in general you can't tell the difference between the two formats. One scenario where you might want to use 64-bit audio is when a signal is going to be heavily processed through a long chain of plugins. By using 64-bit audio you can reduce the accumulation of rounding errors that might occur by using a less precise number format.

Cantabile supports both 32 and 64 bit audio as a global option - see Options → Audio Engine → Double precision (64-bit) audio.

Note that not all plugins support 64-bit audio - check the plugin's documentation.

## Understanding Cantabile's Multi-processor Support

Cantabile supports multi-core/processor machines in the following ways:

- The audio engine runs on a separate thread to the user-interface.
- Many tasks performed by the audio engine are delegated to a pool of worker threads to help spread the load.
- Many background tasks are performed on background threads (audio decoding, writing audio recordings, audio buffer management etc...)

### What's a Thread?

On this page, and in any discussion of multiple-process support, the term "thread" is often used. A thread like a sub-program within a program that performs some task.

The key thing about threads is that the operating system can schedule threads to run on different CPU cores and is the underlying principle for leveraging multi-core machines - by running two computationally intensive tasks at the same time, rather than one after the other, the total time taken should be less.

### Parallel Processing

The main thing to understand here is that multiple core's don't help when process a task whose input depends on the output of a previous task.

For example, say you have two plugins - an instrument, followed by an effect. There's no point processing the effect on a separate core because it can't start it's work until the instrument has been processed. In fact, in this case the additional cost of switching between threads would make the process *slower*.

On the other hand, if you have two instruments they can be processed on separate threads since neither needs the output of the other for its input. If they're processed on separate threads, the operating system can schedule these to different CPU cores and they can both be processed at the same time.

### Why Doesn't Cantabile use 100% of all cores

If you run Cantabile under heavy load with many plugins you'll notice that rarely will it bring CPU load to 100% on all cores. Why is this?

If you think about the above example of two instruments being processed in parallel, it's extremely unlikely that both instruments will take exactly the same amount of time to process their audio - different plugins have different processing load.

In other words, Cantabile needs to pause at certain points while it waits for slower plugins to finish before it can continue - and it's these pauses that cause CPU load to never reach 100%.

Of course the other reason for never hitting 100% is unrelated to multi-processor support and is due to the fact that everything is being processed faster than the duration of the audio cycle - so everything pauses until the next cycle. In fact this is the ideal - and typical - situation.

### Understanding Cantabile's Multi-Processor Option

During early development of Cantabile's multi-processor support we noticed that some plugins don't work correctly if two instances of the same plugin are processed at the same time. (eg: two instances of the same instrument plugin)

For this reason Cantabile has a couple of options (Options → Audio Engine) that controls multi-processor support. The available options are:

Number of Audio Threads:	1 - disable multi-core support	▼
Multi-processor Mode:	Aggressive	▼
Thread Affinity Mode:	Forced	▼

## Number of Audio Threads

Controls the number of threads Cantabile will use to perform audio processing. Normally this should be set to the number of physical processors your machine has. For processors that support hyper-threading it should be the number of actual physically processor cores - not the number of virtual hyper cores.

By setting the number of audio threads to 1 you can effectively disable Cantabile's multi-core support. This might be useful if a plugin proves to be completely incompatible with the multiprocessor support. Generally this mode shouldn't be needed but may be useful for diagnostics reasons (particularly for plugin developers) or for some unique scenarios this might actually provide better performance (eg: when only using a single plugin).

By setting this to a number lower than the number of physical processors available you can leave processing power available to other programs, or for plugins that support their own built-in multi-core processing.

## Multi-Core Mode

Compatibility Mode provides significant performance increases when running most multi-rack songs and is the recommended mode for most situations. In this mode racks are processed in parallel but processing will stall if two or more plugins of the same type need to be processed at the same time - in which case they will be processed one after the other.

Aggressive Mode is suitable when running many racks with the same plugins on each rack. In this mode the plugins being used must be compatible. Many plugins are compatible with this mode, but those that aren't can cause undesirable effects ranging from noise to crashing the entire application.

When a song contains no duplicate plugins Compatibility Mode and Aggressive Mode are effectively equivalent.

## Thread Affinity Mode

This option can be used to force Cantabile's audio threads to run on the same core. This can provide some performance advantages but should be tested on a system-by-system basis. (most systems will show negligible difference).

# Understanding Cantabile's Performance Metrics

## Time Load

Cantabile's time load display is a measure of how long it takes to process one audio cycle. It's calculated as a fraction of the length of the audio buffer.

eg: if you audio buffer is configured to be 10ms and it takes 4ms to process the audio cycle, load will be reported as 40%.

Cantabile displays the maximum load of all audio cycles in the previous 1 second.

Note that this is a measure of processing time, not CPU load - which are quite different things. It's possible for time load to be high but CPU usage to be low. For example, reading data from disk is slow and takes time but CPU usage is low (since it's just waiting). Cantabile also displays CPU load metrics however these are less useful for assessing real-time audio performance.

With a well written host and plugins, CPU load and Time load should correlate - but are measured in very different ways and can not be compared.

Sometimes it's possible for load to exceed 100% without audio drop outs. This can happen if one audio cycle is stalled but the average is still well below 100%. In this case additional audio driver buffering can compensate for the one slow cycle. If however load is continuously, or excessively above 100% audio drop outs will result.

# jBridge

## Use x86 plugins in x64 Cantabile

jBridge is a technology that makes it possible to load x86 (32-bit) VST plugins into x64 hosts. Cantabile 2.0 has integrated



support for jBridge and will automatically use it if it is found to be installed.

For more information about jBridge, please see <http://jstuff.wordpress.com/jbridge/>

Please note:

- jBridge was not developed by Topten Software - it's an independant technology developed by João Fernandes.
- For support enquiries and information on known issues relating to jBridge please see the jBridge documentation.

Special thanks to João for allowing the use of this technology in Cantabile.

## Using jBridge with Cantabile

To use jBridge with Cantabile, install it using the installer available from <http://jstuff.wordpress.com/jbridge/>. Once installed Cantabile will detect its presence and automatically bridge plugins as required.

- It is not necessary to create or rename DLL files as described in the jBridge documentation.
- After installing jBridge, you may wish to update your plugin path to include folders containing the plugins for the alternate platform and re-scan the plugin folders.
- Bridged plugins are labelled in Cantabile's main window with an "(x64)" or "(x86)" suffix.

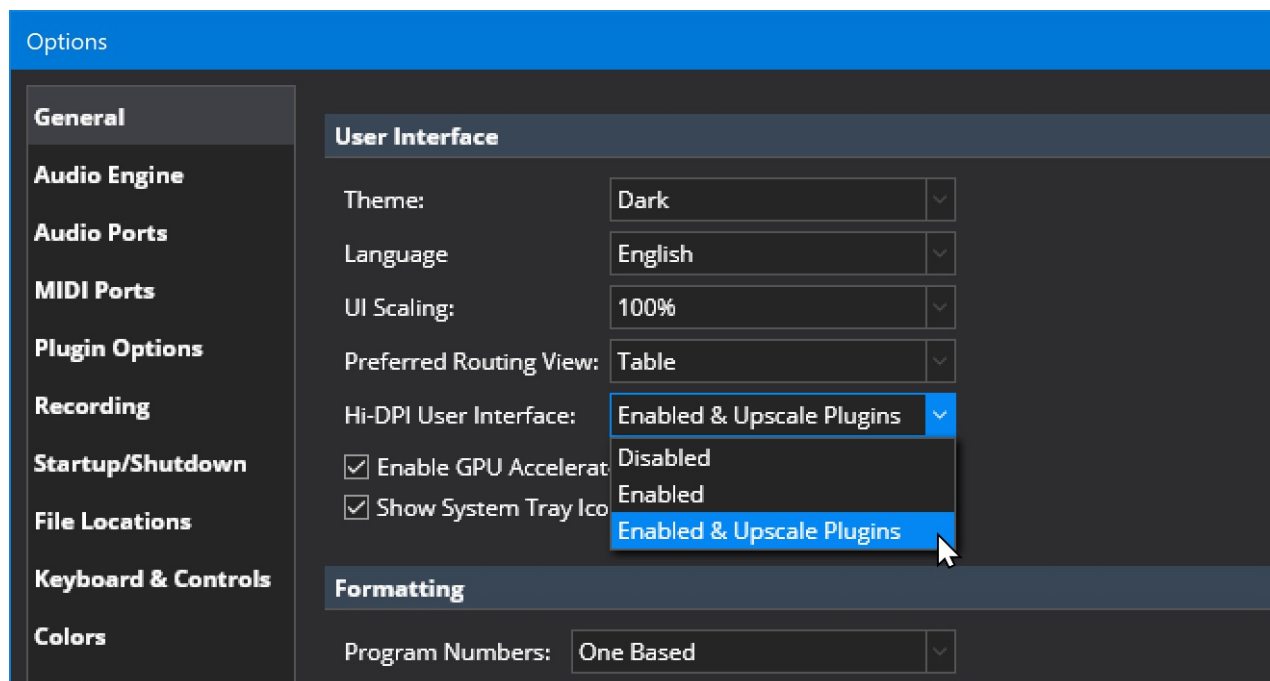
## Support for High-Resolution Displays

Cantabile supports high-resolution display devices however since many plugins don't support this there are a couple of options to control this, depending on which version of Cantabile you're running and which operating system.

### Earlier Versions

In versions of Cantabile before build 3500, or if running on Windows editions before Windows 10 Creators Update there is a single option to enable or disable high-resolution support in Cantabile.

In Options → General → User-Interface → HiDPI User Interface:



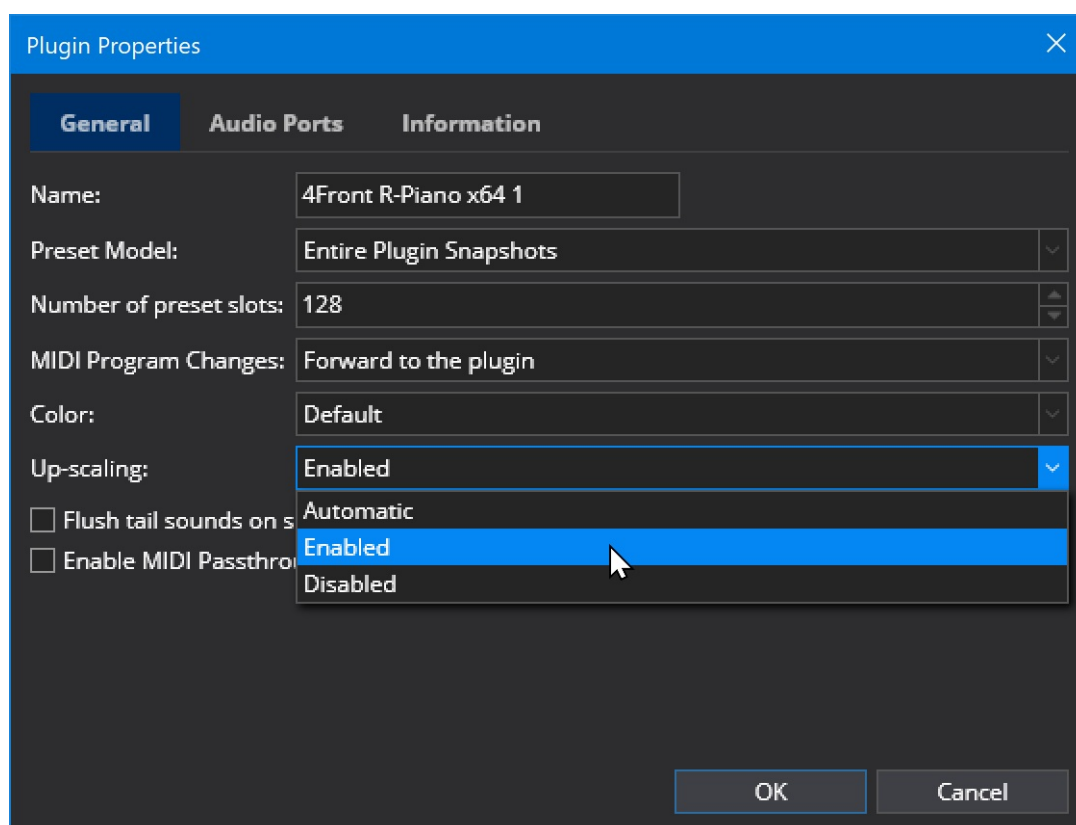
### Windows 10 Creators Update and Cantabile 3500 and Later

Cantabile build 3500 and later can take advantage of new features in Windows 10 Creators Update to support "Per-Window Resolution Scaling" and allows Cantabile's main window to be high-resolution while the plugin editors remain at low resolution and correctly sized.

This feature is optional and off by default because it's unclear at this stage the impact this might have on the stability of plugins.

To enable this feature ensure "Enabled & Upscale Plugins" is selected in the above HiDPI setting.

If "Enabled & Upscale Plugins" is selected, each kind of plugin must also have upscaling enabled. To configure this, insert an instance of the plugin, right click on it and choose "Properties...". In the properties page you'll find an option "Upscaling":



The following options are available:

- Automatic - VST 2 plugins are upscaled, VST 3 plugins are not
- Upscaled - Cantabile will attempt to upscale the plugin (not all plugins work)
- Disabled - no upscaling will be applied, check the plugin's documentation to see if it has its own scaling options

## Understanding Cantabile's Licensing System

Cantabile's licensing system has been designed to be simple to use while at the same time providing the protection from piracy that any modern software business requires to survive.

For more on the background rationale behind Cantabile subscription model, see [this page](#).

### About Activation

Cantabile uses a product activation scheme where each installation requires registration with the Cantabile Software website.

Unlike other product activation schemes, Cantabile's licensing is not designed to restrict properly licensed users from installing on multiple machines. Rather it's intended as a mechanism to stop the runaway piracy that occurs when a stolen serial number or license key is public posted to hacker forums.

Activated installations don't need to be de-activated - so long as you're within the terms of the [license agreement](#) (which is very liberal) you should never encounter issues with extra installations.

For information on what the license agreement entitles you to, see [Installation on Multiple Machines](#)

### Online Activation

Online activation is the easiest way to activate Cantabile and is almost trivial to use:

- Start Cantabile
- You'll be prompted to enter you email address and license key
- Enter the license key you received via email
- Choose which license to use

## Offline Activation

Offline activation should be used when the machine you're installing Cantabile on doesn't have internet access:

- Start Cantabile
- Press the "Offline Activation" button
- You'll be shown a code. Copy it down.
- Visit this page: <https://www.cantabilesoftware.com/offline>
- Login if necessary
- Enter the code you copied down above
- Choose which license(s) you'd like to activate
- Download the generated license file and transfer it to the target machine
- Double click the file to install it
- Back in Cantabile click the Continue button and choose the license to use

## Switching Between Licenses

Often you'll have multiple licenses installed. For example you'll often have a license for the free Cantabile Lite, Trial licenses for Solo and Performer as well as any purchased licenses.

You can switch licenses by choosing the "Change License" command from Cantabile's Help menu.

## Your Cantabile Account

Your Cantabile Account let you view your licenses, view receipts for purchases, purchase upgrades and to facilitate offline activation.

You can access the your account in any of the following ways:

- By visiting [this page](#)
- By clicking the "Account" link in the top right corner of any page on the [cantabilesoftware.com](https://www.cantabilesoftware.com) website
- By choosing "Cantabile Account" from Cantabile's Help menu

To access the portal you'll need to set a password. If you haven't yet set a password and need to access the portal you can do one of the following:

- Use the portal link in the original email you received when registering to use Cantabile
- Use the Cantabile Account command from Cantabile's Help menu with a valid license installed. Cantabile will automatically log you in using your license key for credentials.
- Use the Forgot Password link on the login page.

## Backing Up Licenses

If you're concerned about needing to reinstall Cantabile when internet access is not available you can back up either the download license files used during offline activation, or the file where Cantabile stores the installed licenses:

C:\Users\<<yourusername>>\AppData\Local\Topten Software\Cantabile3.licenses

Note that license activations are tied to the product key of the Window installation - so to use the same licence activations on a different machine you'll need to use the same Windows product key.

(To be clear: we don't have access to your Windows product key - we create a one-way hash based on it)

## About Upgrades

Licensed Cantabile Solo users can purchase an upgrade to Cantabile Performer for the difference in price between the two editions. See below for how to purchase an upgrade.

## Purchasing Upgrades and Subscription Renewals

Upgrades and subscription renewals can be purchased through the User Portal:

- Login to your [Cantabile Account](#). See above.
- Click to expand the license you want to purchase the upgrade or subscription renewal for.
- Links will be shown for any available upgrades - click the link and follow the instructions.

## Understanding Cantabile's Subscription Model

When you purchase a license for Cantabile Solo or Performer you also get a 12 month subscription for one year worth of free updates. Beyond that first year there is a annual subscription to continue to receive updates.

This page clarifies some information about the subscription model and also explains the rationale behind it.

## It's Not a Subscription to Use the Software

The first thing to say about this pricing model is that this is not a mandatory annual subscription to use the software. The subscription is only for updates.

If your subscription expires, the software will continue to run—you'll just miss out on any new features and updates.

## Why Subscriptions?

During the development of Cantabile much thought went into ways to ensure it can survive as a long-term viable business. An important part of that viability is some recurring revenue.

Most software companies earn this recurring revenue by releasing a new version every year or so and charging an upgrade price. This model is very common and generally well accepted by customers.

The problem with this model is the way it incentivizes the software company. In this model, the developer is incentivized (possibly even required) to hold back new features until next year's version. The motivation is to have a nice long bullet list of features to make the upgrade attractive to customers.

Compare this to an update subscription model. In this model the developer is motivated to provide a steady stream of worthwhile updates—lest the customer perceive the subscription as not worth the subscription fee and cancels.

There's one other reason I've decided on this model: Cantabile's development depends heavily on a fast feedback cycle between implementing new features and then fine tuning them over time. A subscription model provides much better support for this kind of development.

## Regular vs Early Bird Subscriptions

When you renew your subscription:

- If your subscription hasn't expired, your new subscription will add 12 months to the end of the current expiration date (so you get a full year worth of updates) and you get an early bird discount of about 20%.
- If your subscription has expired, your new subscription will cover you for 12 months from the current date (you still get a full year worth of updates) but you won't get the early-bird discount.

## My Subscription Has Expired but I Don't Want to Renew

If your subscription has expired and you don't want to renew it you can still continue to run older builds which are available from the Release Notes ([v3](#) [v4](#)) page. Just find the last build released before your subscription expired and that build will run fine with your current license.

You can find the date your subscription expires by logging into your account and clicking on your purchased license to view its details.

## How to Renew Your Subscription

To renew your subscription:

1. [Login to your account](#)
2. Click on the license whose subscription you want to renew
3. Click the "Extend Subscription" button and follow the instructions
4. Once you've renewed your subscription you can [download the latest version of Cantabile](#) and Cantabile will automatically re-activate itself (if required).

## Installation on more than one PC

With Cantabile's licensing we've tried to reach a balance between user convenience, protecting our own investment and what is generally deemed to be "fair".

The implications and requirements for installation on multiple PC's falls into two categories - Legal and Technical.

### Legal

Cantabile's license agreement allows installation on multiple PCs - so long as only one instance is used at a time.

Some common scenarios:

- one computer for home practice and one for live performance - only requires one license.
- a rack of PC's all running Cantabile at the same time - requires a license for each one.
- two PC's for live performance, an active one and a hot standby machine - only a single license is required.

See also: [Cantabile License Agreement](#)

## Technical

In technical terms, Cantabile's licensing system uses a product activation scheme where all installations are registered with the Cantabile Software website.

Unlike other product activation schemes, Cantabile's licensing is not designed to restrict properly licensed users from installing on multiple machines. Rather it's intended as a mechanism to stop the runaway piracy that occurs when a stolen serial number or license key is public posted to hacker forums.

Activated installations don't need to be de-activated - so long as you're within the terms of the license agreement (which is very liberal) you should never encounter issues with extra installations.

*We hope that our customers will honour their obligations to ensure they're correctly licensed for their particular situation.*

## Unsure

If you're unsure on how the licensing terms apply to your particular situation, please [contact us](#) and we'll be more than happy to discuss it.

## My serial number doesn't work

The most common reason for a serial number not working is that it's not been entered correctly. Wherever possible use copy/paste to enter the both the licensee name and the serial number - both of these must be exactly as issued. One character wrong and it will not work.

If you still can't get your serial number to work, please [contact us](#) - preferably including a screen-shot of how you've entered your details.

## Sys-ex Patterns

Sys-ex Patterns are used to extract data from and to generate MIDI System-Exclusive messages (aka: "Sys-ex" messages).

MIDI sys-ex messages consist of a sequence of bytes formatted in a manner specific to the hardware manufacturer. The MIDI standard defines several standard sys-ex messages but hardware manufacturers are also free to define their own formats.

Rather than build in sys-ex support for every possible piece of hardware, Cantabile uses sys-ex patterns as a flexible way to extract data from just about any fixed length sys-ex message. They can also be used to generate sys-ex messages.

Currently the use of sys-ex patterns is limited to use in the Sysex Encoder and Sysex Decoder [MIDI Filters](#) - see below for more details.

A sys-ex pattern consists of a sequence characters that define the format of a sys-ex message. The following describes the syntax of these patterns.

## Hex and Binary

The simplest form of a sys-ex pattern is an exact, byte for byte representation of the sys-ex data.

eg:

F0 00 00 00 F7

This would match (or generate) the exact 5 bytes shown where each byte is hex encoded. Note that sys-ex patterns must always include the leading F0 and trailing F7 bytes that are part of the MIDI standard.

You can also use binary encoding by enclosing the binary digits in square brackets.

eg: this is identical to the above.

[11110000] 00 00 00 F7

Binary digits can't cross a byte boundary so the following are illegal:

```
0[11110000]
[1111000011110000]
```

## Whitespace

White space between bytes is optional - but must be between bytes. Whitespace in the middle of a byte will generate an error:

This is valid:

```
F0000000F7
```

This is not:

```
F 0000000F7
```

Nor is this:

```
F0 [00] [111111] 00 00 00 F7
```

## Wildcard Characters

Sometimes when matching sys-ex data you might not care about the value of a particular byte. In these cases you can use '?' in place of any digit:

This:

```
F0 00 ?? 00 F7
```

Would match any of these:

```
F0 00 00 00 F7
F0 00 10 00 F7
F0 00 01 00 F7
```

But not this:

```
F0 10 00 00 F7
```

You can also use wildcards for one half of the byte:

```
F0 00 0? 00 F7
```

and in binary numbers:

```
[00??00??00]
```

If you use a wildcard character in a pattern used to generate sys-ex data it will be replaced with zero.

## Variables

Variables can be used to capture bits from a sys-ex message being decoded or to substitute bits into a sys-ex message being generated.

This sys-ex pattern will capture the 3rd byte into the val variable (where it can be used later in generating a standard controller message).

```
F0 00 $(val) 00 F7
```

When generating sys-ex data the opposite applies and \$(val) will be substituted with the data from the variable.

You can also capture (or generate) a sub-set of bits from a byte in a sys-ex message using bit ranges. Bit ranges are specified using a high and low bit number separated by two periods (..). Bit zero is the least significant bit and bit 7 would be the most significant bit in a byte.

eg: [3..0] means bits 3 to 0

eg:

```
F0 00 0$(val[3..0]) 0$(val[7..4]) F7
```

The above pattern would capture:

- The lowest 4 bits from the third byte into the lowest three bits of the `val` variable
- The lowest 4 bits from the forth byte into the highest three bits of the `val` variable.

eg:

```
F0 00 [0]$(val[6..0]) 00 F7
```

would capture the lowest 7 bits of the third byte into the lowest 7 bits of the `val` variable. Note the leading `[0]` to keep correct alignment.

You can also capture a single bit using `$(val[5])` (for example).

Note that the order of the bit numbers in a range is important and must always have the higher bit number first. `$(val[4..7])` is not valid.

## Sys-ex MIDI Filters

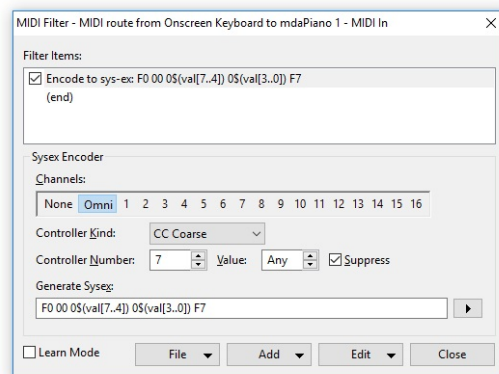
The sys-ex encoder and decoder MIDI filters use these patterns to convert sys-ex messages to and from other standard MIDI messages such as CC and NRPN/RPN messages.

In both of these filters the following variables are available:

- `$(ch)` - the MIDI channel number
- `$(cc)` - the MIDI controller number or (N)RPN parameter number
- `$(val)` - the value of controller or (N)RPN parameter

## Sys-ex Encoder Filter

The sys-ex encoder filter takes a standard MIDI controller message and generates a sys-ex message.



The filter looks for incoming MIDI messages with the following settings:

- MIDI Channel - which MIDI channels to monitor
- Controller Kind - the kind of incoming MIDI message to convert
- Controller Number - the controller number (or parameter number for (N)RPN messages) to match, or "Any" to match any
- Controller Value - the controller value to match, or "Any" to match any value

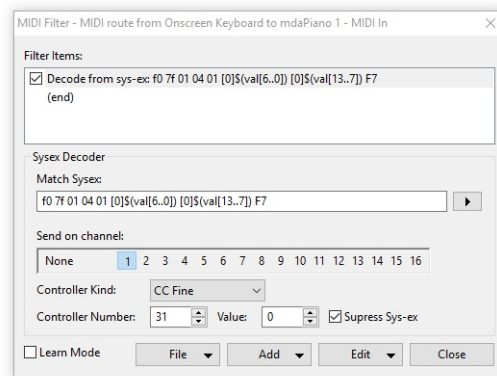
Once matched, the values from the matched message are stored in the variables described above and the sys-ex pattern is used to generate a sys-ex message.

If the Suppress option is turned on, the original MIDI message is suppressed otherwise both the original message and the sys-ex message are sent.

In the above example, the filter is only matching on CC #7. You could however change this to "any" and then use the `$(cc)` in the sys-ex pattern to include the cc number in the generated sys-ex message.

## Sys-ex Decoder Filter

The sys-ex decoder filter is essentially the opposite of the the encoder filter:



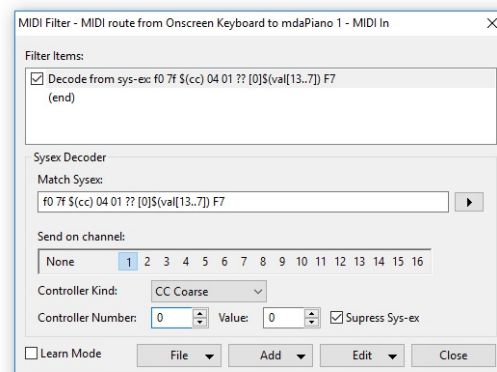
This filter looks for an incoming sys-ex message that matches the supplied pattern and updates the variables from the sys-ex data. Once a matching sys-ex message is found a MIDI message is generated using the supplied settings.

Generally for each variable you'll want one of two behaviours:

1. Always use a specific value - in which case set the value in the MIDI filter and don't include the variable in the sys-ex pattern. Or...
2. Use a value from the sys-ex data - in this case, leave the MIDI filter setting at zero and use the appropriate variable in the sys-ex pattern.

The example above is matching on a the standard MIDI sys-ex for master volume, extracting the 14-bit value from bytes 6 and 7, storing them in the \$val variable and then sending the value as 14-bit MIDI CC 31.

Suppose instead of always sending on CC 31 you want to extract the CC number from the device number of the sys-ex message. Also, let's also suppose you only wanted the coarse 7-bit value:



In this example we're:

- Extracting the controller number from byte 3 (\$(cc))
- Ignoring the fine part of the value (??)
- Set the Controller Number back to zero

Note that setting the controller number back to zero is important. Without that the value from the sys-ex data and the supplied value will be bit-wise ORED together.

The Suppress Sys-ex option causes the matched sys-ex message to be suppressed. When turned off, both the sys-ex and the generated controller message are sent.

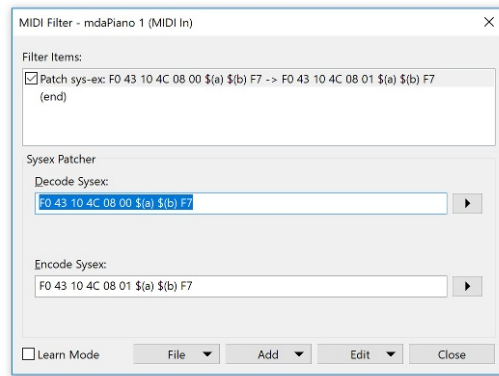
## Sys-ex Patcher MIDI Filter

The sys-ex patcher filter provides a way to generate a similiar sys-ex event from an incoming sys-ex event.

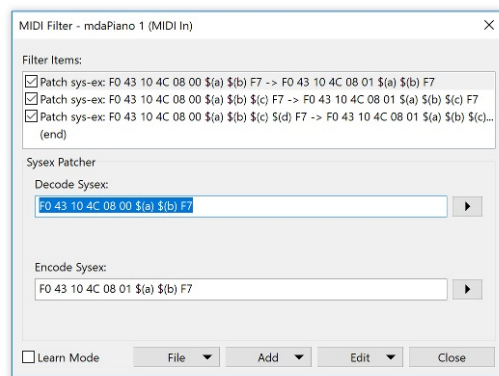
To use the sys-ex patcher MIDI filter you give it two sys-ex patterns. One which decodes the incoming sysex and if it matches, uses the second pattern to generate a new sysex message. Variables can be to pass arbitrary data from the decode pattern to the encode pattern. You can have have as many variables as you like (well up to 256) and you can name them what ever you like.

The following example, uses two variables \$(a) and \$(b) to capture two values from the incoming sys-ex message and generate a new sys-ex message with just one byte changed:





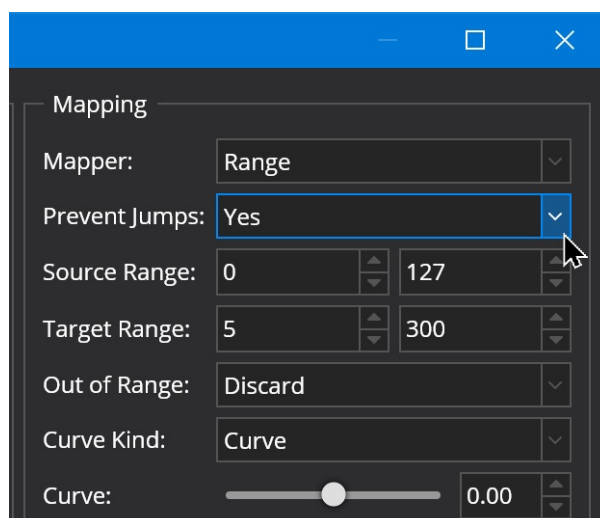
The above will work if the incoming sys-ex event always has the same length. There's no direct support for variable length sys-ex pattern matching. However if you have a very limited range of data lengths you could use a couple of filters. eg: say the data could be 2, 3 or 4 bytes in length you could do this:



## Binding Jump Prevention and Relative (Rotary) Encoder Support

### Jump Prevention

For value to value type bindings, the value mapper supports an option "Prevent Jumps". This is useful when mapping absolute value controllers (eg: regular knobs/sliders) and prevents the binding from causing large sudden jumps to the new controller position.



When enabled, the binding won't have any effect until the incoming value "catches up" to the current target value.

For example, imagine this scenario:

1. You have a song loaded with a slider mapped to the volume of a plugin. During the song the slider is moved to a

reasonably normal volume position.

2. You then switch to your next song which has the same slider bound to the volume of another plugin, but initially the plugin's volume is set to silent and you want to use the slider to fade-in the volume of the plugin.

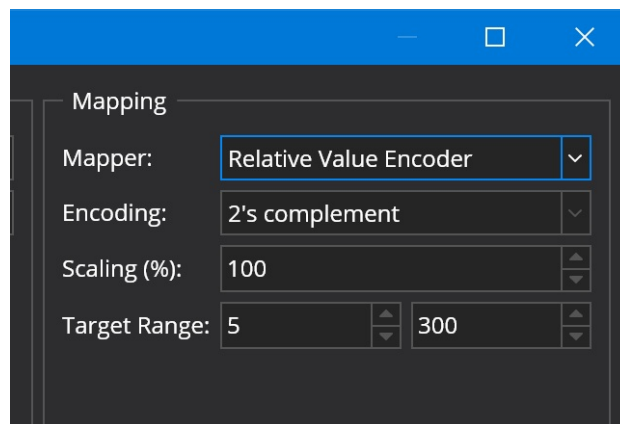
At the point where the second song is loaded the position of the slide and gain setting of the plugin in the new song will be out of sync (the plugin turned right down, the slider positioned in a louder position).

- Without jump prevention, moving the slider a small amount would cause the plugin gain to instantly jump to the loud position.
- With jump prevention, the volume of the plugin won't be affected until the slider is brought down to the current gain setting of the plugin at which point it catches and then starts updating the gain of the plugin.

## Rotary Encoders

Rotary encoders (also called relative encoders) are a special kind of rotary knob that are endless. Instead of sending an absolute position, rotary encoders send relative movements (ie: turned clockwise or turned anti-clockwise).

To support rotary encoders, Cantabile has a special "Relative Value Encoder" mapper:



The relative encoder modes are supported:

Two's Complement

standard [two's complement](#) signed encoding (1 = +1, 127 = -1)

Offset

0 is the negative most value and 127 is the positive most value

Sign Bit

The highest bit is set for negative values and the lower 6 bits indicate the magnitude

Note:

- refer to the documentation of your MIDI device to determine which mode to use (or just experiment with different modes until you find the one that works)
- by default most MIDI controllers with rotary encoders send absolute CC data and often you'll need to reconfigure the knob to send relative mode data.
- the relative encoding modes match the same settings in Reaper.

## Relative Mode Scaling

By default the relative steps are configured so that 127 steps of a coarse controller will cover the full target range, or for fine controllers 16383 steps.

You can adjust the speed at which the controller affects the target value with the Relative Scaling property. eg: setting it to 50% will make the target value move at half the speed.

Note that you can set a negative scaling value to reverse the direction of movement.

## Incrementing and Decrementing Values with Buttons

The relative mode options allow configuring two buttons to increase and decrease a value.

For example, suppose you have two buttons on CCs 16 and 17 and that each sends the value 127 when pressed. You could configure these to adjust a parameter in the following way.

This binding will increment the value:

Edit Binding - Bindings - General - #1

Source

Learn...

Object: MIDI Ports

Point: Main Keyboard (in)

Event: Controller

Controller: 16 Ch: 1

Routing Mode: Continue

Target

Object: Master Levels

Point: Output Gain

Mapping

Mapper

Encoding

Scaling

Target

Comments

☐ ☒ Enabled ☐ Bi-directional

Timing...

Test

then use a second binding with a negative scaling to decrement the value:

Edit Binding - Bindings - General - #2

Source

Learn...

Object: MIDI Ports

Point: Main Keyboard (in)

Event: Controller

Controller: 17 Ch: 1

Routing Mode: Continue

Target

Object: Master Levels

Point: Output Gain

Mapping

Mapper

Encoding

Scaling

Target

Comments

☐ ☒ Enabled ☐ Bi-directional

Timing...

Test

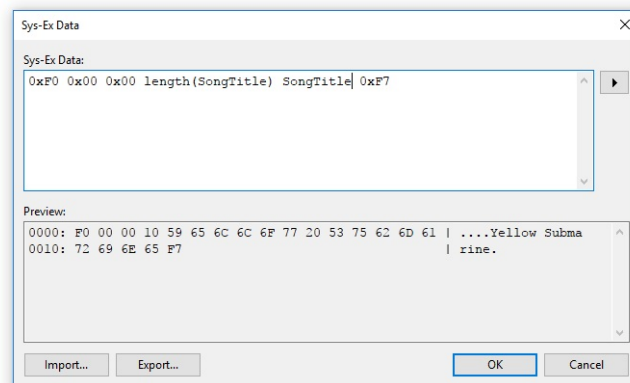
Note that the decrement controller has a negative scaling and both bindings have the relative scaling set to 3% to scale down the 127 value sent by the buttons down to a more finely grained adjustment of the target value.

If you can configure the value sent by the buttons you can use just one binding and have one button send a positive value and one a negative value (according to the relative mode you have selected).

## Sys-ex Expressions

Sys-ex Expressions are used to generate MIDI System-Exclusive messages (aka: "Sys-ex" messages) from a [binding](#).

Sys-ex expressions can consist of a simple fixed series of data bytes or can be used to dynamically generate sys-ex messages based on the source value of a binding.



## Accessing the Sys-ex Expression Editor

Sys-ex expressions are used in bindings where the target side of the binding is a MIDI Sys-ex. To enter the sys-ex data, create a binding where the Target is a MIDI port, the action is "Sys-ex" and then click in the second half of the action column:

State	Source	Event	Target	Action
● ✓ ▶	Song	On Load	0ms Output Port - External Synth	- SysEx x"F7000000F0"

## Syntax

A sys-ex expression consists of a series of individual expressions which are evaluated to generate the final sys-ex message. The following data types and syntaxes are supported:

- A simple decimal value. eg:

123 => 7B

- A hex value prefixed by 0x eg:

0xF7 => F7

- An math expression evaluating to an integer value: (see below for a list of supported operators)

(10 + 20) \* 2 => 60 decimal => 3C

- Floating point math is also supported, but will be converted to integer value in the final sys-ex data:

0.5 \* 10 => 05

- Strings (which will be ascii encoded in the final byte stream)

"Hello" => 48 65 6C 6C 6F

- Arrays (which will be flattened to an array of bytes)

[ 1, 2, 3 ] => 01 02 03

- Sys-ex Patterns (a more convenient way to express a series of hex values). [See here](#) for more.

x"F7000000F0" => F0 00 00 00 F0

- Variables

SongTitle => "Yellow Submarine" => 59 65 6C 6C 6F 77 20 53 75 62 6D 61 72 69 6E 65

- Functions

substr(SongTitle, 0, 6) => "Yellow" => 59 65 6C 6C 6F 77

## Element Separators

Normally you don't need to include separators between each element in an sys-ex expression:

```
`0xF7 0x00 0x00 0x00 0xF0` => F7 00 00 00 F0
```

However you can use semicolons or commas if you prefer:

```
`0xF7,0x00,0x00,0x00,0xF0` => F7 00 00 00 F0
```

Or if necessary eg:

```
`0xF7 100 -20 30 0xF0` is interpreted as `0xF7 100-20 30 0xF0` => F7 50 1E F0
```

What you probably want is:

```
0xF7 100, -20 30 0xF0
```

Or,

```
0xF7 100 (-20) 30 0xF0
```

*(That said negative numbers aren't really supported in sys-ex data like this so it's moot point)*

## Element types

The result of evaluating each element in the sys-ex expression must be one of the following types:

- A number between 0 and 255 (ie: a byte). Values outside this range are considered an error
- A string which will be ASCII encoded
- An array in which each element must be a number between 0 and 255, a string, or another array.

The final byte stream is constructed by flattening and concatenating the result of all expression elements.

## Local Variables

You can use local variables to store intermediate values during the evaluation of the sys-ex expression.

eg: this takes the first 8 letters of the song name, wraps it in "--" delimiters, converts it to an ASCII encoded byte array and stores the byte array in a variable called "shortName".

```
var shortName = ascii("-- " + substr(SongTitle, 0, 8) + " --");  
0xF7 0x00 0x10 length(shortname) shortname 0x00 0xF0
```

and will generate sys-ex data similar to the following:

```
0000: F7 00 10 0E 2D 2D 20 59 65 6C 6C 6F 77 20 53 20 | ..... Yellow S  
0010: 2D 2D 00 F0 | --..
```

Note that the 0E byte is the length of the shortName byte array.

## Variables

All of Cantabile's [string expansion variables](#) are available for use in a sys-ex expression however they should be used without the normal \$(...) delimiters. ie: use SongTitle not \$(SongTitle).

## The "VALUE" Variable

A special variable named value represents the current source value of a binding.

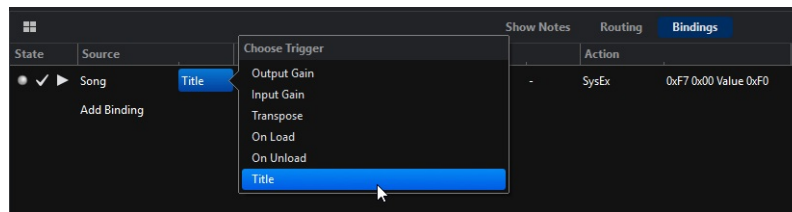
eg: suppose you bound a MIDI CC event to a sys-ex expression you could do something like this:

```
0xF7 0x00 value 0xF0
```

where value would be replaced with the current value of the MIDI controller.

Depending on the source of the binding, the value variable might take on different types.

- For MIDI source values the value is typically an integer value, 0-127 for controllers, 0-16383 for fine controllers etc...
- For audio gain based source values it's a gain multiplier where 1.0 = 0dB
- For string bindings (eg: Song name) it's a string (obviously)
- For key range data bindings it's a byte array (see below)



## Key Range Data Format

The "Engine - Key Range Data" binding sends a byte array for the value variable. The byte array contains three bytes for each key range currently visible on the on-screen keyboard, as follows:

- byte 0 = the color of the key range where 0 = default, 1 = red, 2 = maroon, 3 = green etc.. (see Cantabile's color menus for the rest of the colors)
- byte 1 = the note number of the lowest note in the key range
- byte 2 = the note number of the highest note in the key range

For example, consider the following byte array value:

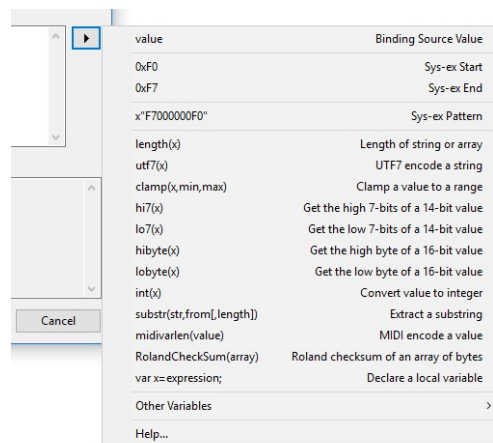
01 00 32 03 35 66

This represents two visible key ranges:

1. 01 00 32 = color 1, key range 0x00 (decimal 0) (C-2) to 0x32 (decimal 50) (D2)
2. 03 35 66 = color 3, key range 0x35 (decimal 53) (F2) to 0x66 (decimal 102) (F#6)

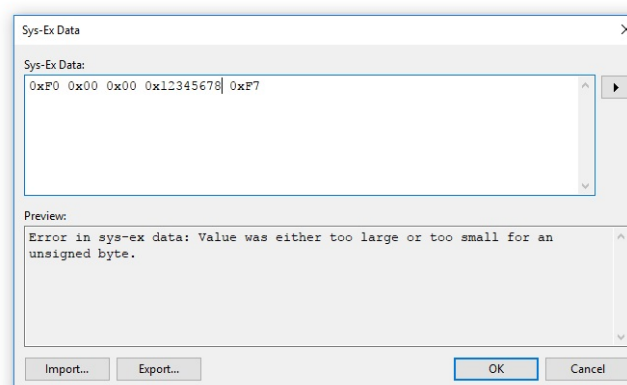
## Functions

See the [Function Reference](#), or the popup menu in the expression editor for commonly used functions:



## Error Handling

If an error is encountered while editing the sys-ex expression it will be shown in the editor output preview area.



If an error is encountered when attempting to send the sys-ex message, a suitable message will be written to Cantabile's

debug log and the sys-ex message won't be sent. You can check the debug log by either:

1. In Tools → Open Settings Folder → look for the file log.txt
2. In Options → Diagnostics, turn on Console Logger to see the error messages as they happen

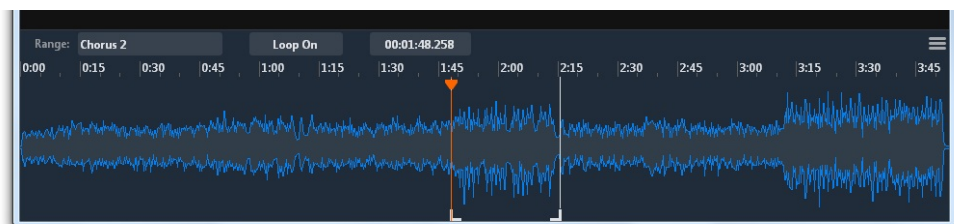
## Operators

The expression engine supports the following operators:

+ Add  
- Subtract and negative  
\* Multiply  
/ Divider  
?: Ternary (aka conditional) operator  
<, <=, >, >=, ==, != Comparison  
! Conditional Not  
&& Conditional And  
|| Conditional Or  
( ) Grouping  
~ Bitwise Not  
| Bitwise Or  
& Bitwise And  
^ Bitwise Xor  
<< Bit-Shift Left  
>> Bit-Shift Right

## Timeline Panel

Cantabile's Timeline Panel provides a detailed view of the contents of audio and MIDI files and can be used to set play ranges and loop modes.



## Showing and Hiding the Timeline Panel

To show the onscreen keyboard:

- Choose Timeline from the View menu, or
- Press Ctrl+E.
- Hold the shift key while dragging up from the edge of the main window, or from the top of the on-screen keyboard.

To hide the timeline panel:

- Resize it (by dragging it's top border) until it disappears.
- While the timeline is active, press *Shift+Escape*.

To leave the timeline visible, but move focus back to main panel press *Escape*.

## Multiple Media Players

The timeline panel shows the contents of the selected file in the most recently activated media player. If you have multiple

media players in a song you can select which file is shown in the timeline panel by selecting the associated media player.

## Elements

The timeline panel consists of the following elements:

- Range Selector - shown as "Chorus 2" in the above screen shot can be used to switch between different user-defined ranges
- Loop Mode Selector - choose to enable loop mode on the selected range.
- Position Indicator - shows the time position of the selected indicator.
- Menu - three bar menu on the right hand side provides additional commands.
- Time Axis - shows time range of the displayed part of the file.
- Waveform and MIDI Display - a visual representation of the media file.
- Current Position Indicator - vertical orange bar with handle at the top.
- Left/Right Markers - white vertical bars with handles at the bottom.

## Setting a Play Range

You can set the play range by either:

- Clicking on the left/right markers handles and dragging
- Playing the media file and pressing the [ or ] keys to set the left or right marker to the current play position
- Moving the current play position while playback is stopped and using the [ or ] keys to set the markers.
- Holding the Alt key and click-dragging a range.
- Selecting one of the left/right markers and using the left/right arrow keys to move it.

## Saving and Recalling Named Play Ranges

Sometimes you might like to save a play range for later recall. To create a saved play range:

1. Set the range you want to save.
2. Press Shift+R, or choose Save Play Range from the Timeline menu.
3. Enter a name for the range and press OK.

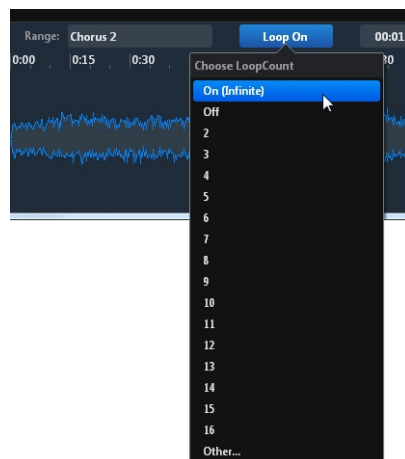
To recall a previously saved range choose it from the play range drop down. You can open this drop down by clicking on it, or pressing the R key.



Each named play range stores the left and right markers, the view zoom/scroll position and the current loop mode setting.

## Setting the Loop Mode

By default playback stops when the end of the play range (or the end of the file) is reached. You can loop playback either indefinitely or a preset number of time using the loop menu:



## Zooming and Scrolling



The timeline can be zoomed the following ways:

- Using the Ctrl+Left/Right arrow keys
- Using Ctrl + Mouse Wheel
- Pressing A to restore the view to the entire file
- Pressing C to zoom to about 15 seconds around the current play position
- Pressing the Z key while clicking and dragging to zoom to a particular range

To scroll the timeline:

- Use the arrow keys, page up/down keys or home/end keys when the waveform or MIDI data is highlighted.
- Use the mouse wheel (either horizontal or vertical mouse wheel will work)
- Use the scrollbar

## Time Format

The displayed position indicator and the timeline axis can be configured to display either musical time (beats and bars/measures) or real-time (minutes and seconds).

To switch format, choose from the timeline panel menu. The selected format is stored with the selected media file so different files can be displayed using different formats.

## Normal, Fine and Coarse Cursor Movement

The cursor and left/right markers can be moved in one of three different granularities. Select the associated indicator by clicking its handle and use the following keys:

- Fine - Shift + Arrow Keys
- Normal - Arrow Keys
- Coarse - Page Up/Down Keys

## Cursor Snapping

When using the mouse to select or drag positions in the timeline it can be configured to snap to aligned boundaries such as seconds, or beats or measures (depending on the current time format).

Snapping can be enabled and disabled in the timeline menu and the operation can be inverted at any time by holding the Shift key while moving the mouse.

## Custom Key Bindings

Many of the shortcut keys for working with the timeline can be customized in Options → Hot Keys.

# Network Developer Guide

This page describes resources of interest to developers working with Cantabile's Network API and Network Server.

## Network API

The network level API allows minimal control of Cantabile using a REST API and full control via a WebSocket interface.

Documentation is [available here](#).

## JavaScript API

The cantabile-js JavaScript library provides a wrapper library for the network API and provides easy control of Cantabile from both Node.JS and the browser.

Documentation:

- [Download/Installation](#)
- [API Reference](#)

## Web Folders

Cantabile's built-in web server serves static content from special .webfolder folders in the [resource folder](#).

A .webfolder can be either a sub-directory of the resource folder, or zipped set of files renamed with a .webfolder extension.

When the web server starts, any `.webfolder` files/folders are mounted using the name of the web folder as the url. For example, everything in zipped web folder `lib.webfolder` would be available as `http://localhost:35007/lib/`

The special web folder `Root.webfolder` is mounted as the root url and contains the default `index.html` file.

Files ending in `.html` can be accessed using a url without the `.html` extension and URLs referring to a directory will server the default file `index.html` if it exists.

## Building Files on Demand

For non-zipped webfolders (ie: actual directories), files can be created/updated on demand by placing a `.make` or a `.bat` file of the same name.

For example, in the `Root.webfolder` the file `styles.css` is generated from the file `styles.less`. By placing a file `styles.css.make` in the same folder, Cantabile will automatically run the make file to generate the dependant file.

eg: `styles.css.make` might contain:

```
styles.css : styles.less
    lessc styles.less styles.css
```

Similarly a `styles.css.bat` could be used for similar purposes. Use of a `.make` file requires a GNU compatible version make is available on the system path (and the above example also assumes `lessc` is available on the system path).

## Built-in Web Content

The built-in web content can be used as a reference and is [available here](#).

Of particular interest might be:

- The default `index.html` page ([see here](#))
- A [standard set of libraries](#) including the `cantabile-js` library, `jQuery`, `Vue`, `Hammer` etc... available as `/lib/`
- A built-in copy of the `Open-Sans` font available as `@import url("/fonts/open-sans/open-sans.css");`