ElectroAcoustic Tools: An Introduction

Version 0.1.1

[What is EAT?](#)
[How do I install EAT?](#)
[What EAT objects are there?](#)
[How do I load EAT modules?](#)
[How do I get fine HID control over parameters?](#)
[How can I save parameter and MIDI presets?](#)
[Will there be more modules?](#)
[Can I provide feedback?](#)

What is EAT?

A package of abstractions for use with Pd-extended, tested on ver. 0.42.5-extended.

How do I install EAT?

Empty the contents of the EAT folder into Resources/extra/unauthorised. Alternatively, a better solution, you may place the folder in the Pd plug-ins folder and identify the EAT folder in the Pd>Preferences>Path… menu item. You can then call any of the EAT abstractions in the same way you would call any native object or external.

What EAT objects are there?

EAT_Audiofile~
Playback of 1–13 channel audio files.

EAT_Delay~
1–13 channels of discrete delays.

EAT_Highpass~
1–13 channels of discrete 1–10 pole high pass filters.

EAT_Lowpass~
1–13 channels of discrete 1–10 pole low pass filters.

EAT_MatrixMix~
13x13 mixer.

EAT_MexicanWave~
Multi-point panner that provides a mexican wave behaviour for 1-12 channels (13 in total including dedicated low frequency channel 13). Contains a 12x12 input mixer with double-click to node default function.

EAT_Mix~
A basic mixer with solo, mute, and level functions for 13 channels.

EAT_Record~
A recorder of audio files.

EAT_Reverb~
1–13 channels of discrete reverbs.

EAT_Transport
Save and load MIDI and parameter presets for EAT modules.

EAT
A shell to house DSP effects.

How do I load EAT modules?

Each abstraction needs some arguments. Every abstraction needs a unique identification tag (e.g. ‘Jeff’) in order to work effectively, some need to be given a number of channels, filters need to be given a pole argument. Here is the syntax and examples for each:

EAT_Audiofile~

Loads audio files up to 13 channels.

Syntax: [EAT_Audiofile~ ID_tag]

Example: [EAT_Audiofile~ Oram]

EAT_Delay~

Request up to 13 channels of delays.

Syntax: [EAT_Delay~ ID_tag *n*-channels]

Example: [EAT_Delay~ Mooney 13]

EAT_Highpass~

Request up to 10-pole filters and up to 13 channels of filters.

Syntax: [EAT_Highpass~ *n*-pole ID_tag *n*-channels]

Example: [EAT_Highpass~ 10 Moore 13]

EAT_Lowpass~

Request up to 10-pole filters and up to 13 channels of filters.

Syntax: [EAT_Lowpass~ *n*-pole ID_tag *n*-channels]

Example: [EAT_Lowpass~ 10 Harrison 13]

EAT_MatrixMix~

Just needs an instance ID.

Syntax: [EAT_MatrixMix~ ID-tag]

Example: [EAT_MatrixMix~ Mclaughlin]

EAT_MexicanWave~

Request a behaviour of up to 12 channels, channel 13 is reserved for LFE.

Syntax: [EAT_Mexicanwave~ ID_tag *n*-channels]

Example: [EAT_Mexicanwave~ Grill 12]

EAT_Mix~

Just needs an instance ID.

Syntax: [EAT_Mix~ ID-tag]

Example: [EAT_Mix~ Juffage]

EAT_Record~

Records up to 13 channels.

Syntax: [EAT_Record~ ID_tag *n*-channels]

Example: [EAT_Record~ Matthews 13]

EAT_Reverb~

Request up to 13 channels of reverbs.

Syntax: [EAT_Reverb~ ID_tag *n*-channels]

Example: [EAT_Reverb~ Steiner 13]

EAT_Transport

No arguments required.

Example: [EAT_Transport]

EAT

Load any EAT DSP effect inside the shell to reduce GPU and CPU consumption.

Syntax: [EAT effect_suffix argument_1 argument_2 argument_3 argument_4…]

Example: [EAT Reverb~ Puckette 13]

Example: [EAT Highpass~ 7 Moore 11]

How do I get fine HID control over parameters?

The sliders are generally a bit small, but you can assign them to any HID sliders or dials that output MIDI 0–127. If you open the 'MIDI' panel (there's one panel for the global controls and one panel for discrete channel control within the ADV window) there are two toggles for each parameter. The ML toggle is MIDI learn for that parameter; the next slider or dial that moves is the control it will learn. Then you can switch MIDI control on or off using the MC toggle.

How can I save parameter and MIDI presets?

You can save your MIDI settings too. If you click on the link at the bottom of any MIDI window to access the transport abstraction, you can save MIDI and parameter presets to a text file. You may want to create your patch within a folder (as if you were running a Pro Tools session, for instance) to ensure the preset files stay together with the right patcher; the text files have generic names, not specific to each patcher containing EAT abstractions.

Will there be more modules?

Yes. This is a small package to see if EAT is useful and to see if the framework is sound.

Can I provide feedback?

Yes. Please provide feedback to richvt@gmail.com putting ‘EAT Feedback’ in the subject line.

Acknowledgments

My thanks go to James Mooney and David G. Thomas to both of whom I am indebted for their generous advice and support during this project. My thanks also go to Scott Mclaughlin for his early testing and criticism. I would also like to thank Thomas Grill, Hans-Christoph Steiner, Thomas Musil, and Miller Puckette whose work has greatly benefitted this project.

You can hear more about EAT: 1100, 9 August 2011, Pure Data Conference, Bauhaus-Universität Weimar, Germany.