ReaperToWwise 1.0 documetation

by Marc Hasselbalch
December 2024

→ What is ReaperToWwise?

ReaperToWwise is a game audio implementation tool that enables you to post events and set game syncs in an open Wwise project from the comfort of the Reaper timeline - without needing to have a game engine running.

You are able to post events, set switches, states, triggers and RTPCs with the use of ***empty items*** by writing text commands in the ***Notes*** field as well as being able to drive RTPCs with automation envelopes.

It's basically a more featurefull, timeline-version of a Soundcaster session.

→ Why does it exist?

The tool was primarily born out of the need for making dynamic Wwise setups for linear videos for use in game audio showreels, to more thoroughly showcase proficiency in Wwise.

Over time it became more and more obvious that making strictly linear audio redesigns for an ultimately interactive medium wasn't doing me much good. But what else do you do when you haven't much in the way of fleshed out and/or released games to showcase? You record some gameplay, remove the audio and redesign the entire sequence from scratch.

There is obviously nothing wrong with this approach, and, needless to say, this is still a dynamic, interactive audio setup for a linear video, but it should better showcase what is needed.

ReaperToWwise will allow you to trigger events and set game syncs in a Wwise project, which would mimic the calls being sent via code from the game engine.

So not only does this tool enable you to better showcase your proficiency in Wwise, but there is also a few other obvious uses, which could be:

1. being able to quickly prototype a setup - or entire systems - in, say, pre-production, before an engine is set up.

2. showcase system functionality to other team members or departments before moving further into implementation

3. spot cinematics or similar sequences

4. and most likely other interesting uses I haven't thought of

→ Where can I download it?

You can download it from ReaPack within Reaper - or directly from the Github repository.

https://github.com/mhasselbalch/ReaperToWwise

→ What is needed to make it work?

ReaperToWwise relies on the dedicated work of others and has a number of pre-requisites. Please read carefully.

Pre-requisites:

Reaper DAW

https://www.reaper.fm/

Note: The tool should be compatible with most Reaper versions, where the Lua interpreter is based on Lua 5.3 (Reaper v.6.x and earlier according to their website).

While Reaper v7 introduced Lua 5.4 functionality, I haven't made use of any of it up until now.

Wwise 2019.1+ (required for installing ReaWwise)

https://www.audiokinetic.com/en/products/wwise/

ReaWwise (can be installed via ReaPack inside of Reaper)

https://blog.audiokinetic.com/en/reawwise-connecting-reaper-and-wwise/

Exposes raw WAAPI functions to Lua inside of Reaper.

ReaImGui (can be installed via ReaPack inside of Reaper).

Responsible for handling the GUI.

→ More details:

ReaperToWwise relies primarily on the Reaper extension called ReaWwise by Audiokinetic, which is described as an extension that *"streamlines the transfer of audio assets from your REAPER project into Wwise. The extension also allows you to create complex object hierarchies in Wwise without leaving the comfort of your REAPER project!"*.

https://www.audiokinetic.com/en/blog/reawwise-connecting-reaper-and-wwise/

A byproduct of ReaWwise is that it exposes raw WAAPI (Wwise Authoring API) functions to Lua inside of Reaper, which is how the communication between Reaper and Wwise is done.

→ Functionality:

The tool works by parsing text commands written inside of the 'Notes' field of empty items.

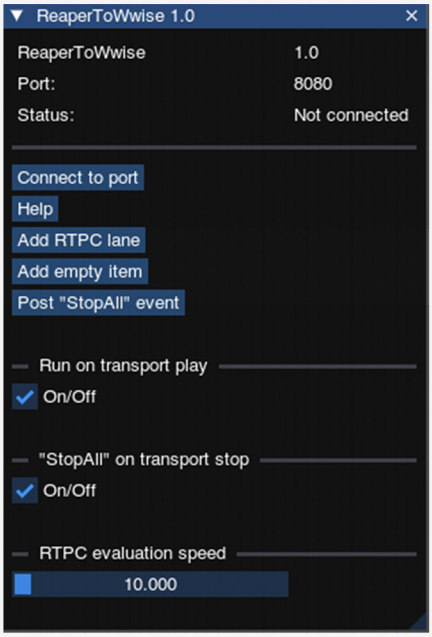
You add an empty item to the timeline, double click it and type in event or game sync call commands and it will be triggered when the Reaper playhead passes the item's start position.

You are also able to drive RTPCs with an automation envelope.

See "The tool window" for all the available commands and RTPC information.

→ The tool window:

Here is a run-through of the elements in the main tool window.



Opens a pop-up window where you can type in the desired network communication port between Reaper and Wwise.

Choose a non-conflicting port.

Check the Wwise network settings at Project → Project Settings → Network → "Game Discovery Broadcast Port (game side)".

Wwise's default is 24024.

The main window status should change from "Not connected" to "Connected" if communcation has been established.

NOTE: If the tool crashes, there is a chance the port will still be occupied even though you close and reload the script. In this case, the tool will auto-input Wwise's default port number.

If everything else fails, close down Reaper and Wwise and try again from scratch.

Opens up a separate window with a few helpful notes on using the tool, including the syntax for the available commands.

The commands are:

Events	e	/	eventName		
Triggers	t	/	triggerName		
Switches	sw	/	switchGroup	/	switchName
States	st	/	stateGroup	/	stateName
RTPCs	r	/	rtpcName	/	rtpcValue

Opens a pop-up window where you type in the name of the RTPC (as it appears exactly in Wwise) and the minimum and maximum values (as you have defined them in Wwise).

A new automation lane is added to the RTPC track at the bottom of the track list and an automatically generated JSFX plugin is added to the RTPC track, which holds a single slider and an automatically created envelope lane for that slider.

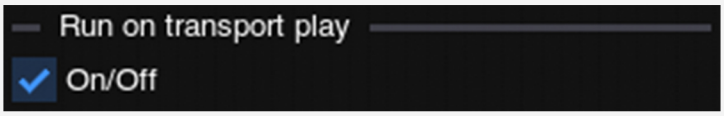
NOTE: You are not easily able to rename an RTPC after the JSFX plugin has been created.

So if you wish to change the name of an already created RTPC, I suggest you add a new RTPC track with the desired name, transfer over the automation data to the new track and remove the old JSFX plugin from the FX list of the RTPC track.

Adds an empty item on the timeline.

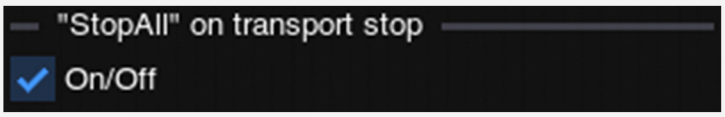
NOTE: You can select a required chunk of time on the timeline and then add the empty item, which will be the length of the selected time. Otherwise it will create an empty item with a considerably too long length.

A manual way to call the “StopAll” event (which is assumed that you have set up in Wwise).
Think of it as a manual panic button.



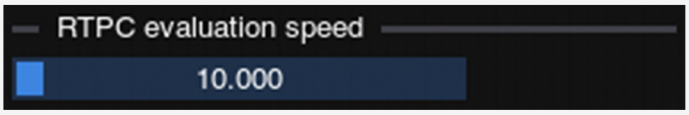
With this checked, the script will run and do all the necessary checks when the Reaper transport is started. It is required for the tool to work.

If you need to go to a different Reaper project and work (and use the transport) before coming back to the ReaperToWwise session, you need to uncheck this, otherwise it will auto-create the RTPC track in the other project, try to look up events and RTPCs and the script might crash, which will require you to open the script again from the desired session.



This will post the event named “StopAll” to Wwise, which is assumed that you have created, when the transport has stopped, to make sure no audio is playing in Wwise once you stop the Reaper transport.

The event should be of the type “Stop All” and be named exactly *StopAll*.
You are free to use the tool without setting this up, but it is recommended for a better workflow.



This slider will, when set to a value more than 10, slow down the polling of values that are sent with the RTPCs to avoid overloading the WAAPI with too much information, which will significantly increase the amount of CPU load from Wwise and slow down both Reaper, Wwise and the rest of your machine.

The default value should in most cases be fine, but you are free to increase this if the need arises

→ Known issues and limitations:

Limitations of timing during tightly sequenced event calls:

> Due to the way I've written the script, very rhythmically tight event calls (such as very rhythmical gunshots or similar) can vary in timing.
>
> This tool does in no way provide sample-accurate event calls as it is based on reading the time position of the Reaper playhead, and that float precision can be difficult to work with.
>
> I've experimented with increasing the float precision and span in which a certain level of imprecision is allowed to still let an event trigger, but I found it be too unstable and would cause the occasional event to be dropped, which shouldn't be an issue with the current configuration.

Help window instances:

> As of now, it is possible to accidentally open up more than one "Help" window. Doing so might result in unexpected behaviour and eventually crash the script when you try to close one of the windows.

Empty item delete upon playback will cause nil reference:

> Please do not delete empty items when the transport has been started, as the script will cause a nil reference and crash.

Thank you to everyone who helped providing support, feedback and tested it.
I couldn't have done it without you.

Thank you to Christian Fillion for providing support and maintaining ReaPack.