Maybe you are experiencing these frustrations

There are endless bugs to fix.
At the last moment before the deadline, a flood of urgent new requirements arrive, and you are asked to work overnight to complete them.

The work sheets, WWUs, and data in Originals for multi-person collaboration are always a mess.

Creating, renaming, dragging, checking, clicking, holding Ctrl, holding Shift, switching tabs, switching back to the previous tab, creating new folders, creating new folders within folders... Endless annoying.

Audio requirements are always given the lowest priority and have to make way for other requirements, but they always lag behind.

Integration always requires configuring a bunch of components, flipping through blueprints, opening editors, drawing circles, building tracks, setting keyframes...

After finally hooking everything up, when submitting the data for comparison, you find that a large number of unknown differences have been generated...

After the test version is finally built, it crashes 9 out of 10 times.

Bug reports always only have a bare video, requiring you to reopen the version, and then it crashed, stuck... Enter countless GM commands, and review countless logs...

Gradually, the desktop becomes cluttered with Excel sheets, PPTs, Word documents, asset libraries, outsourced resources, requirement sheets, integration tools, testing tools... But it seems that there are still a few plugins hidden deep in a folder that were purchased a year ago and used twice, and a MIDI keyboard is still lying in the corner gathering dust...

If you do the math, how many hours a day are spent "being creative" and how many hours are spent on "miscellaneous tasks"?

Where has all the time gone?

Maybe you have long been accustomed to these “invisible miscellaneous tasks” and even feel that “that’s how it should be”

But if you are always rushing around, won't you miss out on too many interesting things in life?

WHIP v2.0 Main Features Introduction

One-click project initialization
For new projects that have not yet generated Wwise data, the plugin includes some preset content that can directly generate an initial Wwise project with just one click! Designers no longer need to spend time creating objects that are common to almost all project types.

One-click integration, mirroring, and starting from scratch
Naming is everything. The moment the designer decides on the naming, a series of processes within the project, from container framework, assigning paths, importing resources to creating Events, and assigning Banks, can be completed automatically with just one click! Designers don't need to waste too much time on "dragging and clicking" (especially not spending time creating many layers of folders locally to apply a certain template structure), allowing them to focus on the refinement of the design itself and dynamic mixing, putting their energy where it matters most.

One-click expand Switch objects
When a new Switch object is added to a Switch group, and accordingly, many related containers and resources also need to be supplemented, the expansion can be completed with a one-click extension of the Switch content. Designers do not need to manually add assignments to a large number of containers one by one.

One-click global safety check
Are there any container objects that are not assigned to an Event? Are there any idle wav files that have been forgotten to be imported into containers? Are there any unused idle Event instances? Are there any objects that do not follow the structural conventions? ... Under the premise that the project structure strictly follows the conventions, designers can obtain a check report of all the above error-prone items with a single click, without worrying about human oversights.

One-click wav relinking
If the Sound's original index is lost due to changes in the local location of the wav file, it can be automatically reconnected with one click!

One-click export of requirement sheet
When you need third-party suppliers to mass-produce samples, you can right-click and export the requirement sheet with one click after selecting the objects that need to be processed.

One-click conversion to worksheet
Under the premise of conforming to format specifications, the Event data of an old project can be exported to a plugin-readable JSON with one click.
The plugin’s JSON can also be converted to xlsx with one click for team collaboration. If the data of the old project is managed in xlsx, it can also be formatted and converted to JSON that can be used by the plugin with one click, allowing the plugin to take over the management!

One-click play, stop, and expand Event
In the corresponding cell of the data table, right-click "Play, Stop, Breakdown Event" to quickly verify sound feedback and view details with one click.
Designers no longer need to go back to the Wwise interface, spend time searching for the Event, click to playback, and so on.

*Interactive Music module is not yet supported in version 2.0

Disclaimer

This Plugin(WHIP v2.0) is only for Testing, Research and Professional Discuss use.
Developer can not guarantee you that there will be no any kind of data loss and data corruption during use.
If you use this plug-in, it means that you agree to bear the risks and possible losses caused during the use of the tool BY YOURSELF, and you DO NOT require the developer to hold any responsibility.

If you fall into any of the following situations, this plugin is likely to be helpful to you:

1. My job is related to game audio, such as "Game Audio Designer", "Game Audio PM", "Game Audio Director", or "Game Audio Programmer".
2. I have used Wwise, understand the basic workflow of Wwise, and am currently using Wwise in my work.
3. My creative environment is demanding, with high daily task pressure and time pressure, and there are no programmers around to assist me. I want to try this plugin to see if it can help me save some time.
4. I want to focus on content design and creation, and don't want to spend too much time on various tools and operational processes.
5. I hope to gain some insights or inspiration in aspects such as "process optimization", "efficiency improvement", "data management", and "tool development" by observing and understanding the form of the plugin itself.
6. I am currently learning game audio-related knowledge and practicing using Wwise, and I hope to get some learning assistance through this plugin.
7. I am following up on a new game project and have sufficient time and opportunity to deploy audio infrastructure work. I want to manage the intricate processes and data clearly and thoroughly in a short period of time.

If you fall into any of the following situations, this plugin may not be suitable for you:

1. My job is not one of "Game Audio Designer", "Game Audio PM", "Game Audio Director", or "Game Audio Programmer", and I have no interest in the game audio industry.
2. I don't use audio middleware similar to Wwise in my work, and I don't plan to learn or understand related professional knowledge.
3. Most of my workload is supported by "complicated manual operation processes", and I worry that automation and AI will weaken or eliminate my value.
4. I don't mind and allow "tedious operation processes" in my daily work, or even rather enjoy "tedious operation processes".
5. The project data I have is already in an irretrievable mess, and I am already exhausted both physically and mentally, and don't want to invest any more energy in so-called maintenance, management, or optimization.
6. I am worried that this plugin may cause any data loss, such as data corruption, data loss, data confusion, etc.
7. I am worried that this plugin may bring any mental loss, such as feelings of pressure, anxiety, loss, fear, and other negative feelings.

Copyright Statement

All files and data related to this plugin (WHIP v2.0), including but not limited to source code and executable programs, have complex copyright components and involve various types of licenses.

Except for "personal research", "non-commercial use, sharing for professional exchange purposes with attribution", the developer is currently unable to directly grant you or publicly provide any authorization for other purposes.

If you intend to use this plugin (including any related data) for other purposes, please be sure to contact me first through the **Developer's Email**. Only after confirming that your intended use does not constitute infringement should you proceed with the usage.

The developer will not be held responsible for any of your usage actions.

Note: In order to contribute to the development of the industry and support the spirit of open-source sharing and mutual assistance, the developer is working hard to advance copyright licensing-related matters. Developer is striving to release the plugin's source code under the most permissive license type possible in the future, allowing interested friends to participate in development! It won't take too long, so please be patient!

Privacy Statement

This plugin (WHIP v2.0) does not rely on the network to run. When you use it, the plugin does not automatically collect any of your private information on your computer.

Developer's Email

whiphelper@gmail.com

WHIP v2.0 User Manual Contents

Compatibility with Different WWISE Versions

*The current version only supports Windows 10 and Windows 11

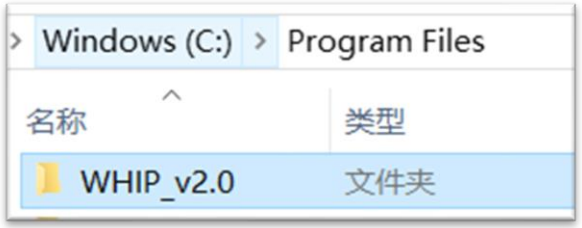
	One-Click Project Initialization	One-Click Integration
2022.1.0 and Higher	**Completely Support**	
2021.1.1 ~ 2021.1.10	Assign RTPC to BUS: NOT SUPPORTED Create Meter Effect: NOT SUPPORTED	**Completely Support**
2019.2.8 ~ 2021.1.0	Set RTPC Value: NOT SUPPORTED Assign RTPC to BUS: NOT SUPPORTED Create Meter Effect: NOT SUPPORTED	**Completely Support**
2019.1.0 ~ 2019.2.7	Set RTPC Value: NOT SUPPORTED Assign RTPC to BUS: NOT SUPPORTED Create Meter Effect: NOT SUPPORTED	Auto Color: NOT SUPPORTED
2018.1.2 ~ 2018.1.11	Set RTPC Value: NOT SUPPORTED Assign RTPC to BUS: NOT SUPPORTED Create Meter Effect: NOT SUPPORTED	Auto Color: NOT SUPPORTED Set Pitch Random: NOT SUPPORTED
2017.2.10 ~ 2018.1.1	Set RTPC Value: NOT SUPPORTED Assign RTPC to BUS: NOT SUPPORTED Create Meter Effect: NOT SUPPORTED The automatic WorkUnits creation is not supported. You must manually create the WorkUnit first. After inputting the path information into the base.json under Wwise Session Path, the subsequent parts can be automated.	Auto Color: NOT SUPPORTED Set Pitch Random: NOT SUPPORTED
Lower than 2017.2.10	**NOT SUPPORTED**	**NOT SUPPORTED**

- **Set RTPC Value not supported**: When the plugin creates a new RTPC object through WAAPI, it cannot directly initialize its maximum value, minimum value, and default value.
- **Auto Color not supported**: When the plugin's security check function identifies problematic objects, it cannot directly modify their icon color in the project through WAAPI to provide designers with intuitive visual prompts.
- **Set Pitch Random not supported**: The plugin cannot directly enable and apply preset Random values to the Pitch of Sound objects through WAAPI.
- **Automatic WorkUnit creation not supported**: The plugin cannot automatically generate WorkUnits and create corresponding WWUs in the project through WAAPI.
- **Assign RTPC to BUS not supported**: The plugin cannot assign RTPC objects to BUSes in the project through WAAPI.
- **Create Meter Effect not supported**: The plugin cannot create Meter objects under the Effects path through WAAPI.

How to Install

*** Method A: For WWISE versions below 2021.1.10, you can only install using the following method: ----------**

Step 1:
Copy the "WHIP_v2.0" plugin folder and place it anywhere on your hard drive (for example, under the C:\Program Files directory).

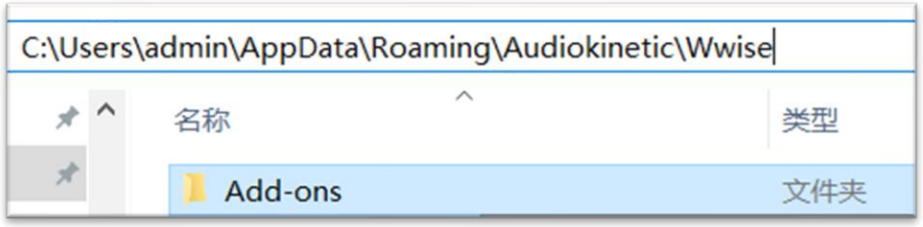


Step 2:
Open the plugin folder, right-click on Launch.exe, and send a shortcut to Desktop. In the future, when using the tool, first open your Wwise session, then double-click this shortcut to launch the plugin.

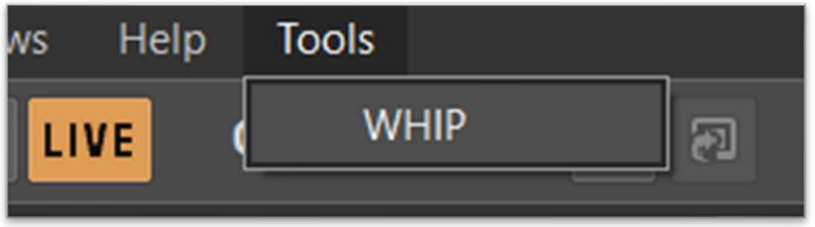
*** Method B: For Wwise 2021.1.10 and higher versions, it is recommended to use the following installation method: ----------**

Step 1:
Place the Add-ons folder in the following path to complete the installation:
C:\Users\[Your User Account Folder]\AppData\Roaming\Audiokinetic\Wwise

Note: If you encounter any conflict prompts during the install process, please cancel the process immediately (do not forcibly overwrite any data), and use Method A for installation instead! If conflicts still occur during the process, please stop all operations immediately and discontinue using this plugin.



Step 2:
After launching Wwise, a "Tools" menu will appear at the top of the menu bar. Click on "Tools" and select "WHIP" from the dropdown menu to launch the plugin.

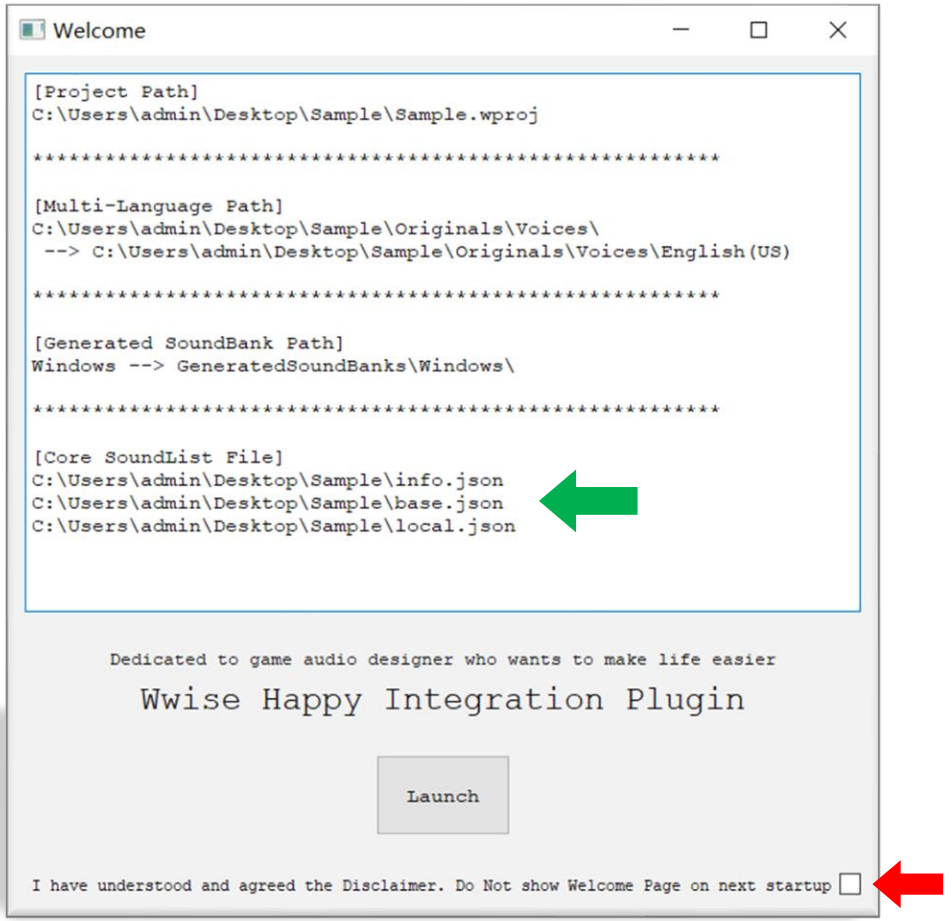


Launch the Plugin (First Time)

First, to ensure data security, the plugin can only run when **there is one and only one Wwise session open**.

If you see the welcome window and the launch button is clickable, it means the plugin has detected the project and **generated 3 essential core JSON files** in the project path, and it can be launched normally.

If you have checked the box in the bottom right corner to hide the welcome window on the first launch, subsequent launches will directly enter the main interface.

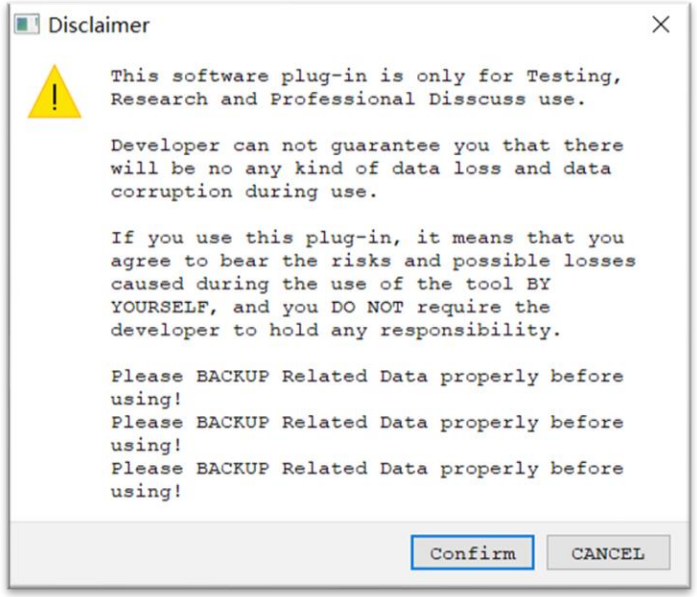


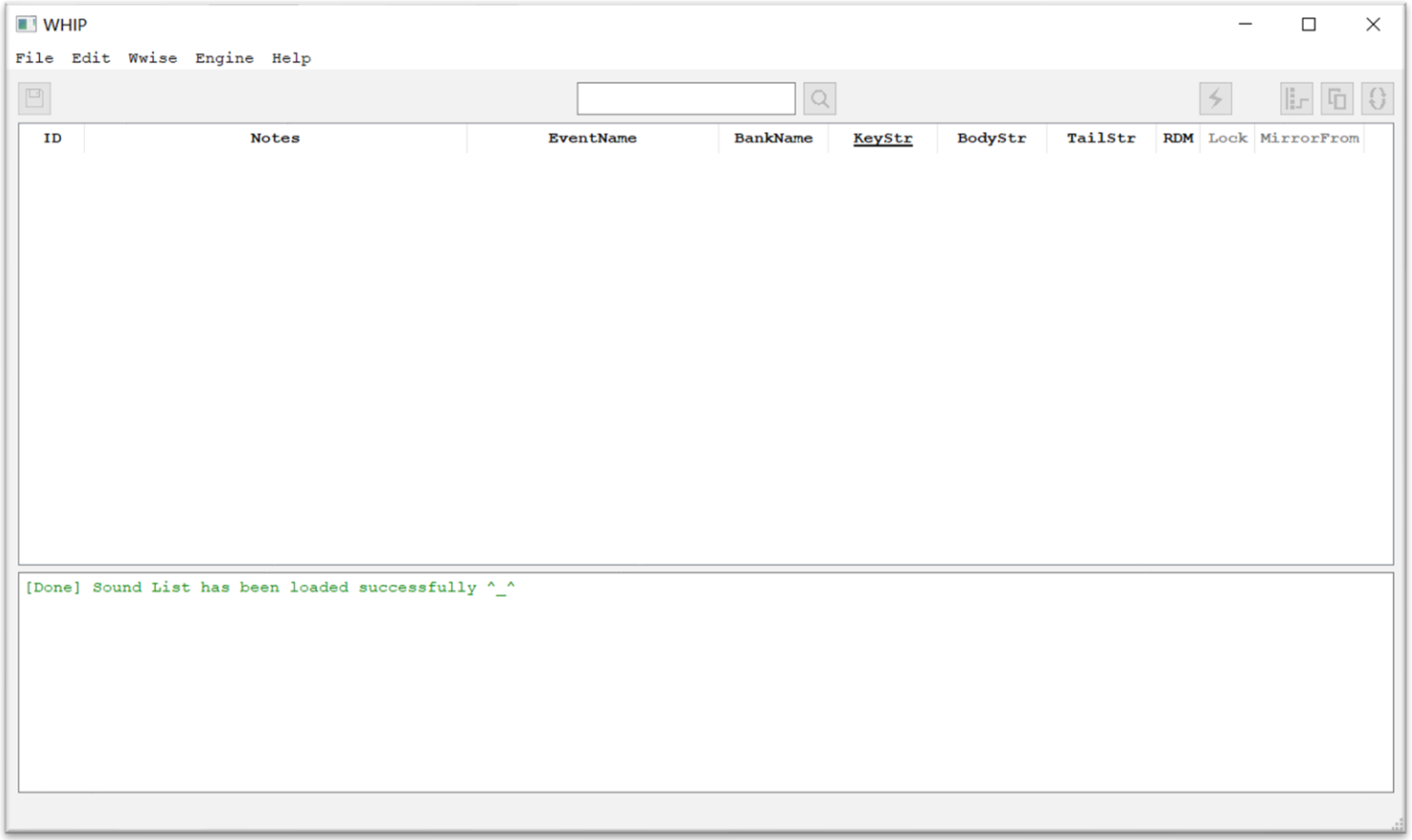
Reminder:

The three files "info.json", "base.json", and "local.json" store most of the data information in the plugin and are the "core" and "essence" of the plugin.

Launch the Plugin (First Time)

After clicking the launch button, you will see a "Disclaimer" prompt.
If you click Confirm button, you will directly enter the main interface of the plugin.

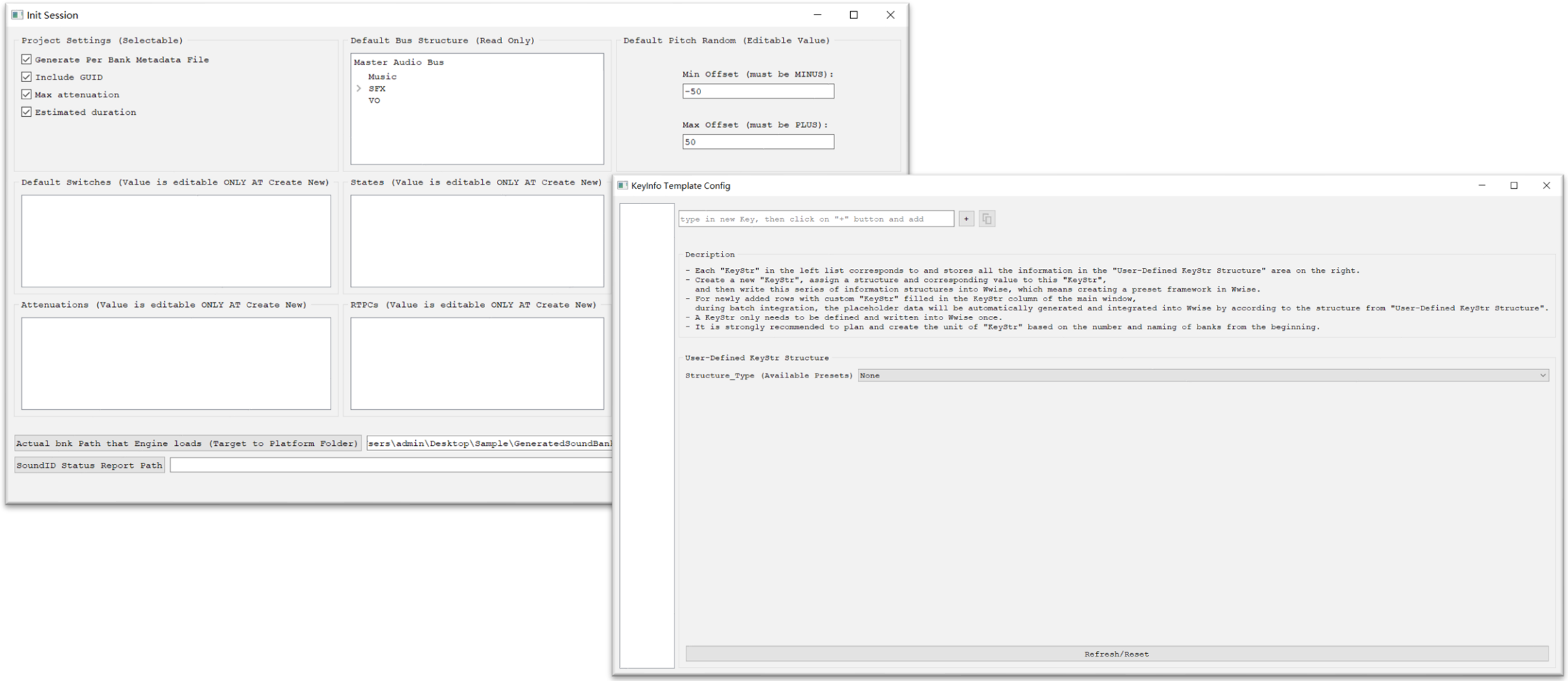




Launch the Plugin (First Time)

In addition to the main interface, the plugin has two other interfaces: Init Session and KeyInfo Template Config.

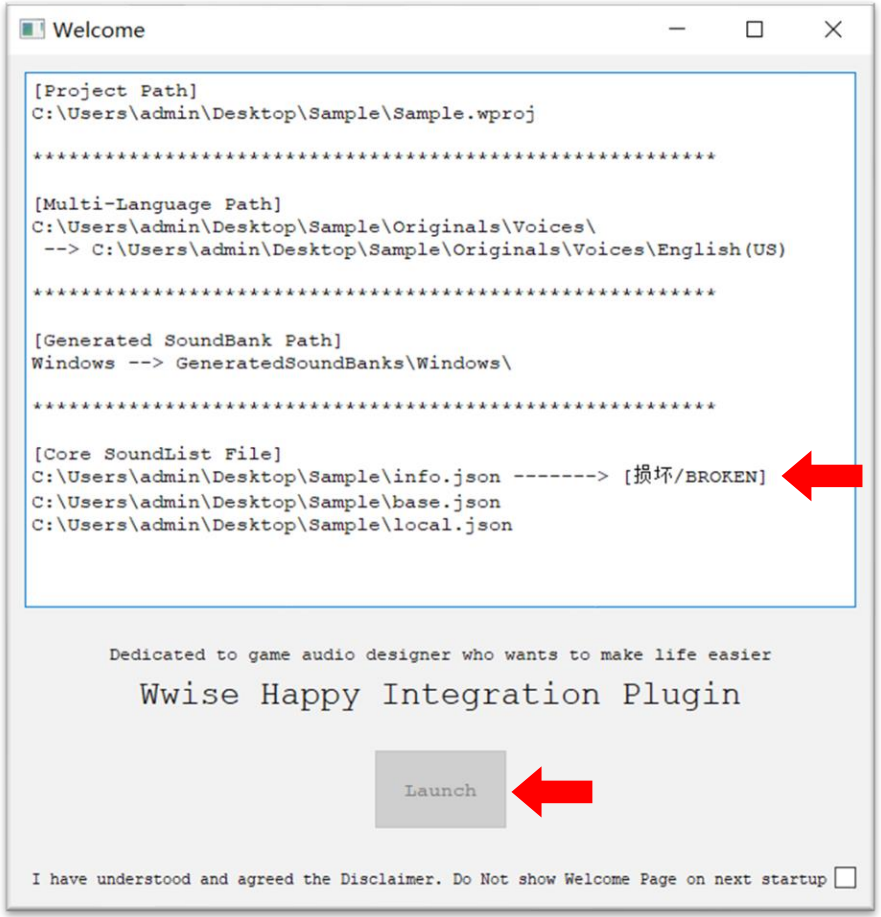
However, these two interfaces are not frequently used. The "Table" in the main interface is the primary operation area of the plugin.



Launch the Plugin (NOT First Time)

If the launch button is grayed out and disabled, it means that the plugin has not found the core audio table files, or the files are corrupted.

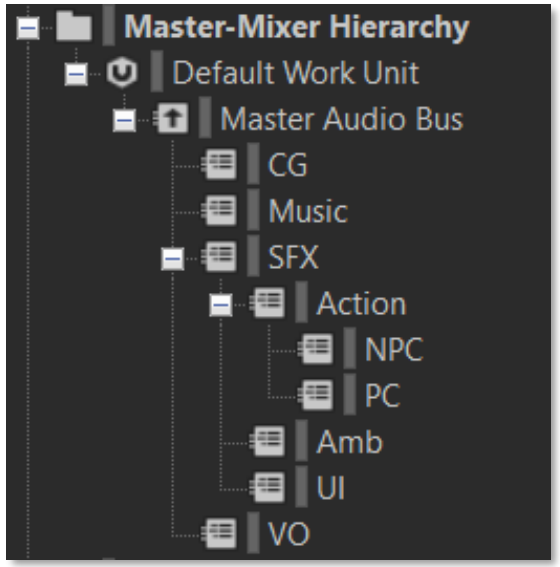
Solution: Repair or delete the corrupted JSON files in the Wwise project path as prompted, and try launching again.

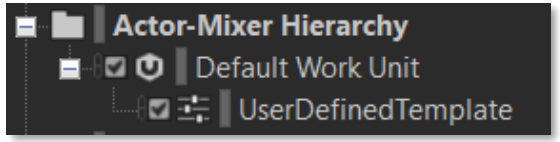


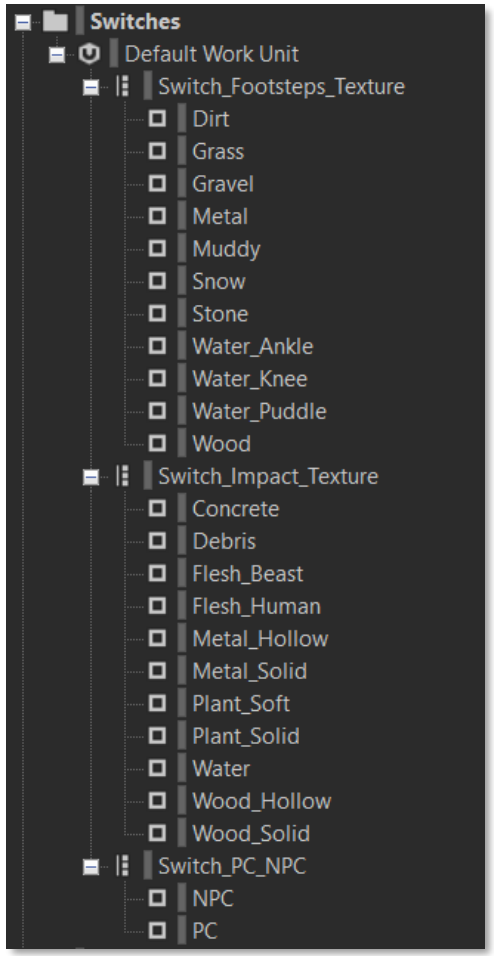
Project Initialization (First Time)

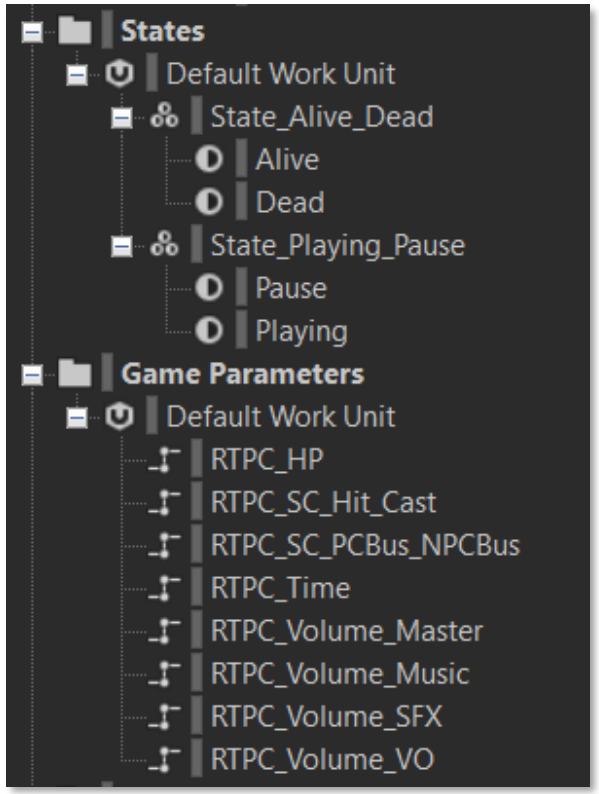
The plugin has prepared some commonly used settings, as shown in the figure below.

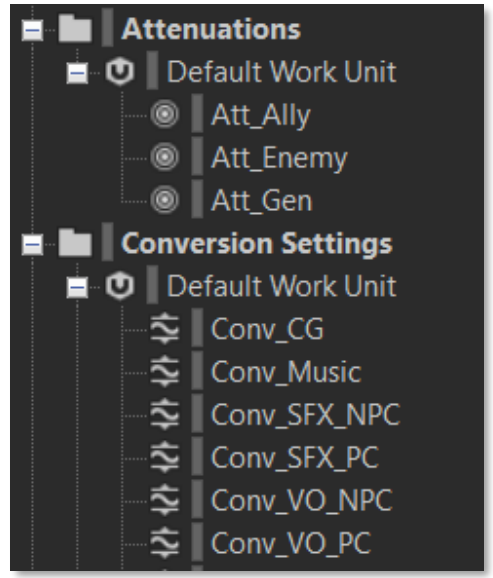
If the current Wwise project is a brand new project, we can select the desired content and write it into the project with one click, saving the trouble of creating them manually one by one.





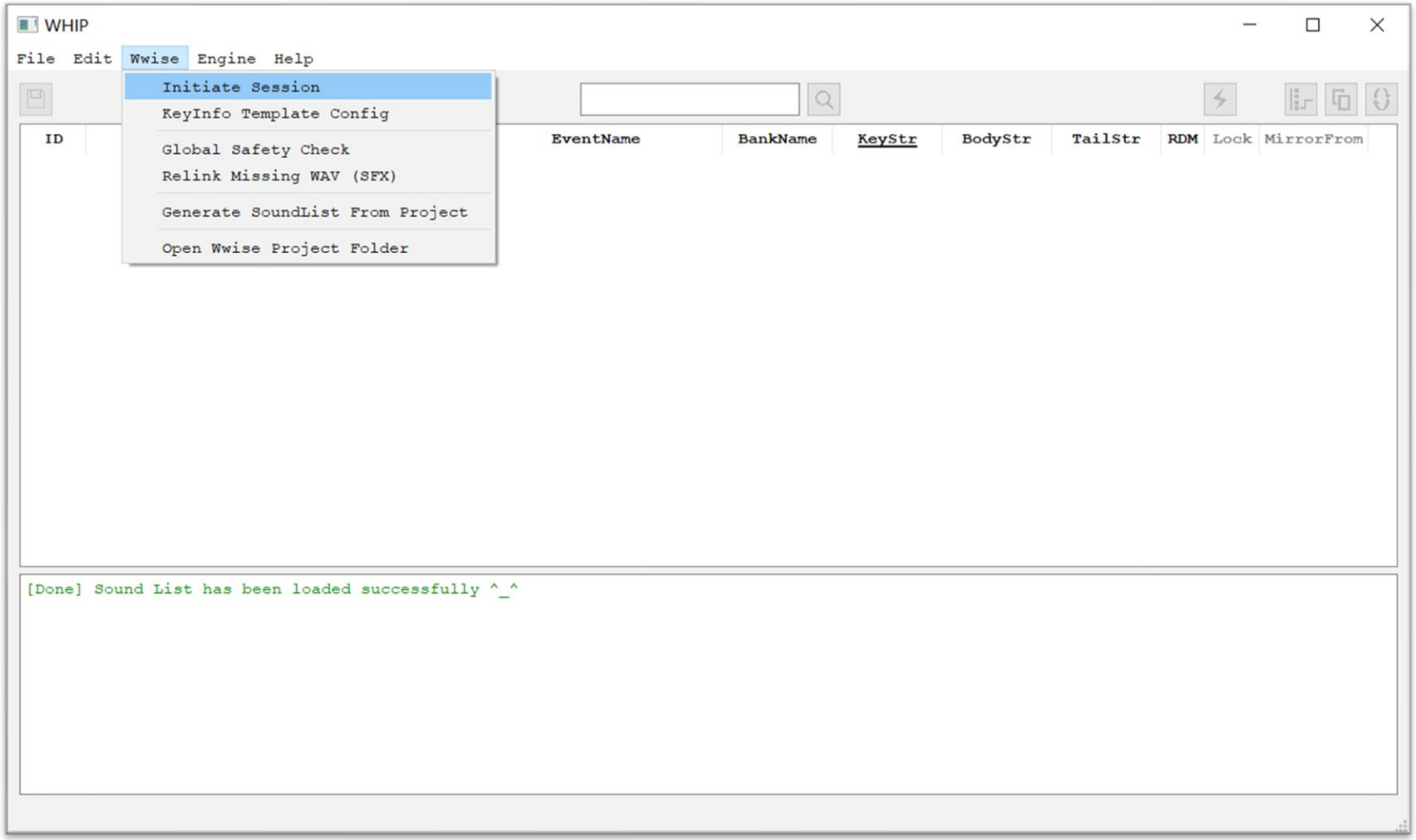






Project Initialization (First Time)

Click on "Wwise" in the toolbar, and then click on "Initiate Session" in the dropdown menu.

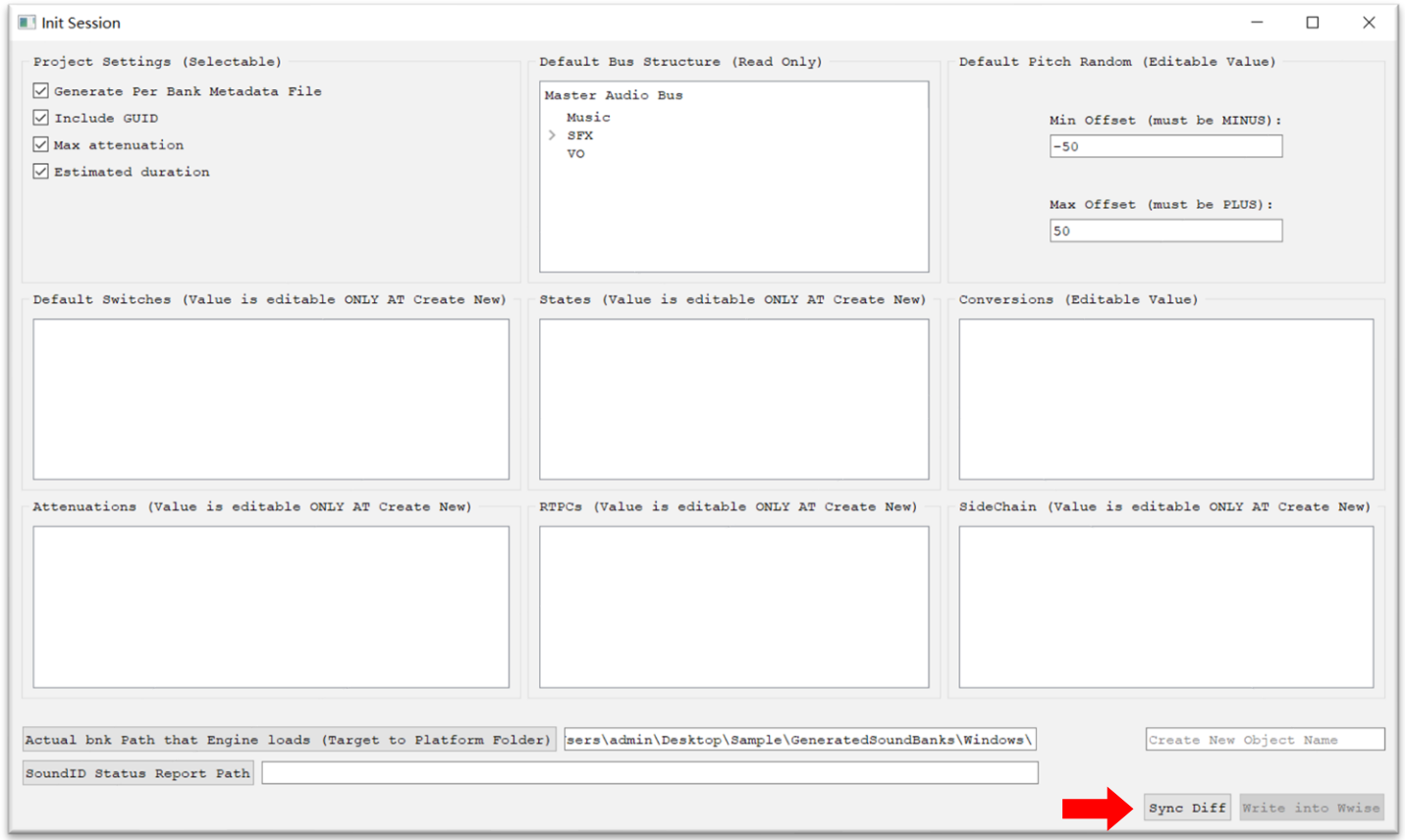


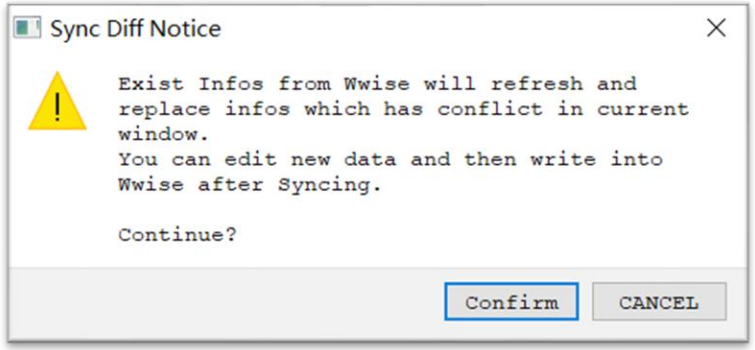
Project Initialization (First Time)

In the Init Session window, there are many module areas.

Before adding content, you need to click the "Sync Diff" button in the bottom right corner to synchronize the actual situation in the current Wwise session to the window.

After clicking the "Sync Diff" button, read the information in the "Sync Diff Notice" pop-up window.
After confirmation, you can start editing the data to be written into Wwise.

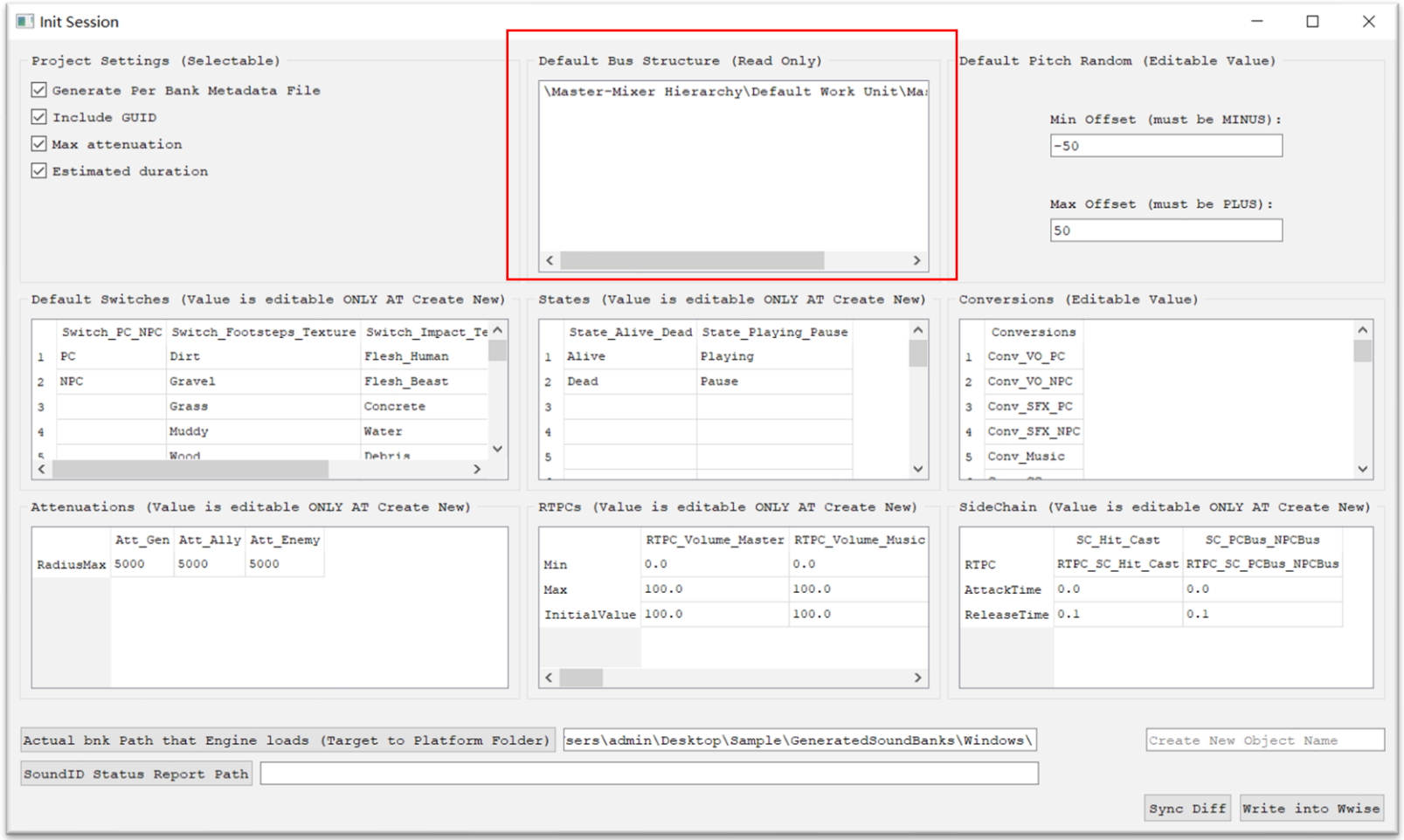




Project Initialization (First Time)

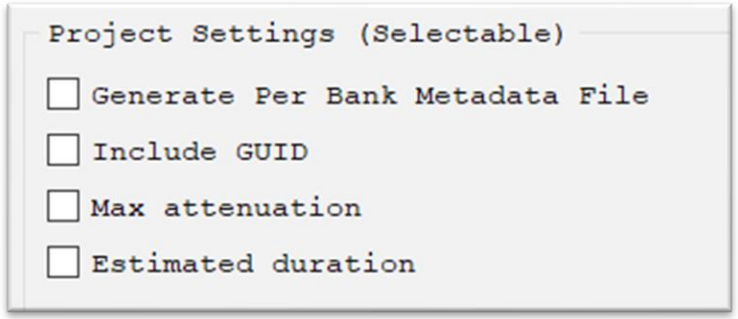
At this point, we can see that the content prepared in advance by the plugin is displayed.

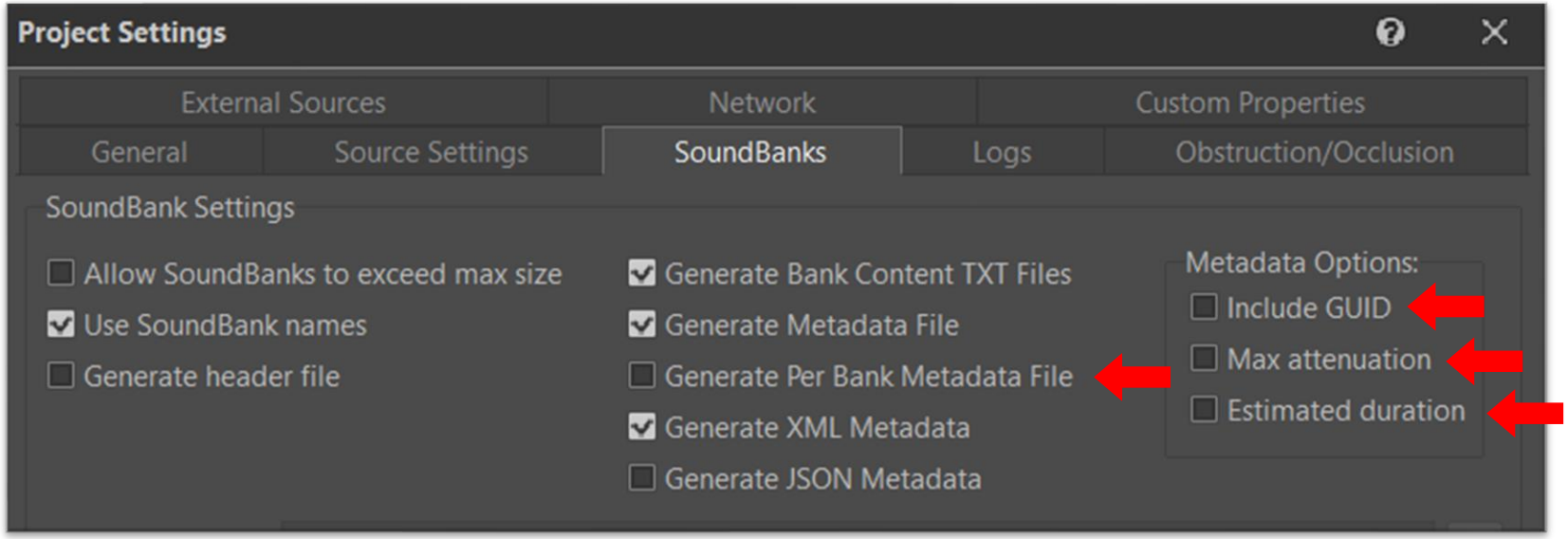
Except for the content in the "Default Bus Structure" area, which is not editable, the content in other areas can be added, deleted, or modified according to your own needs.



Project Initialization (First Time)

The "Project Settings" area corresponds to some of the content in the SoundBanks tab of the Project Settings in Wwise.

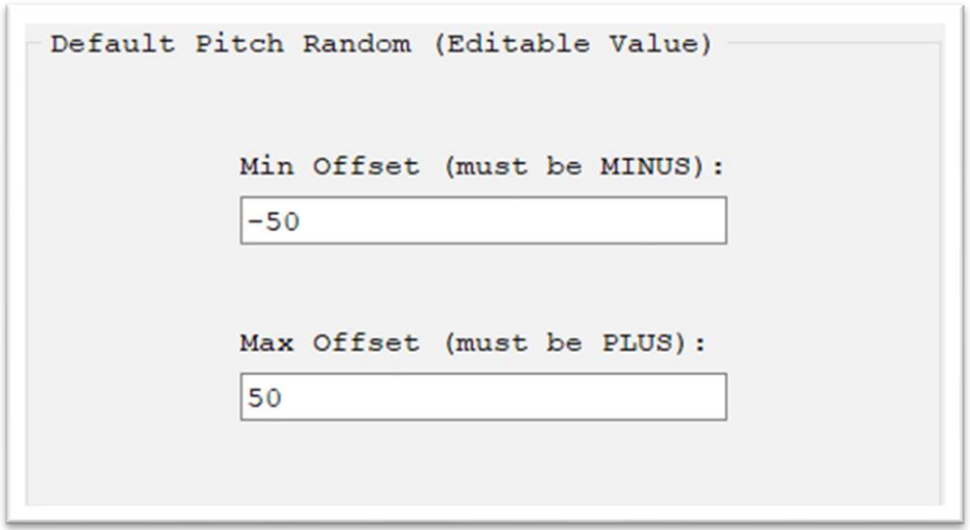


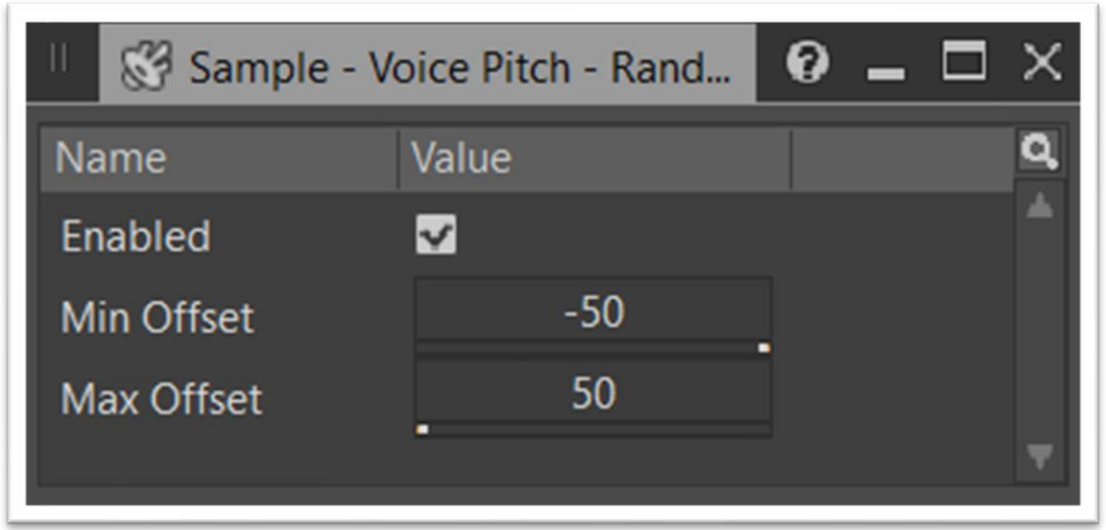


Project Initialization (First Time)

The two values filled in the "Default Pitch Random" area will be used as the default values applied to Sound objects by the automation integration function in the tool.

(Example)





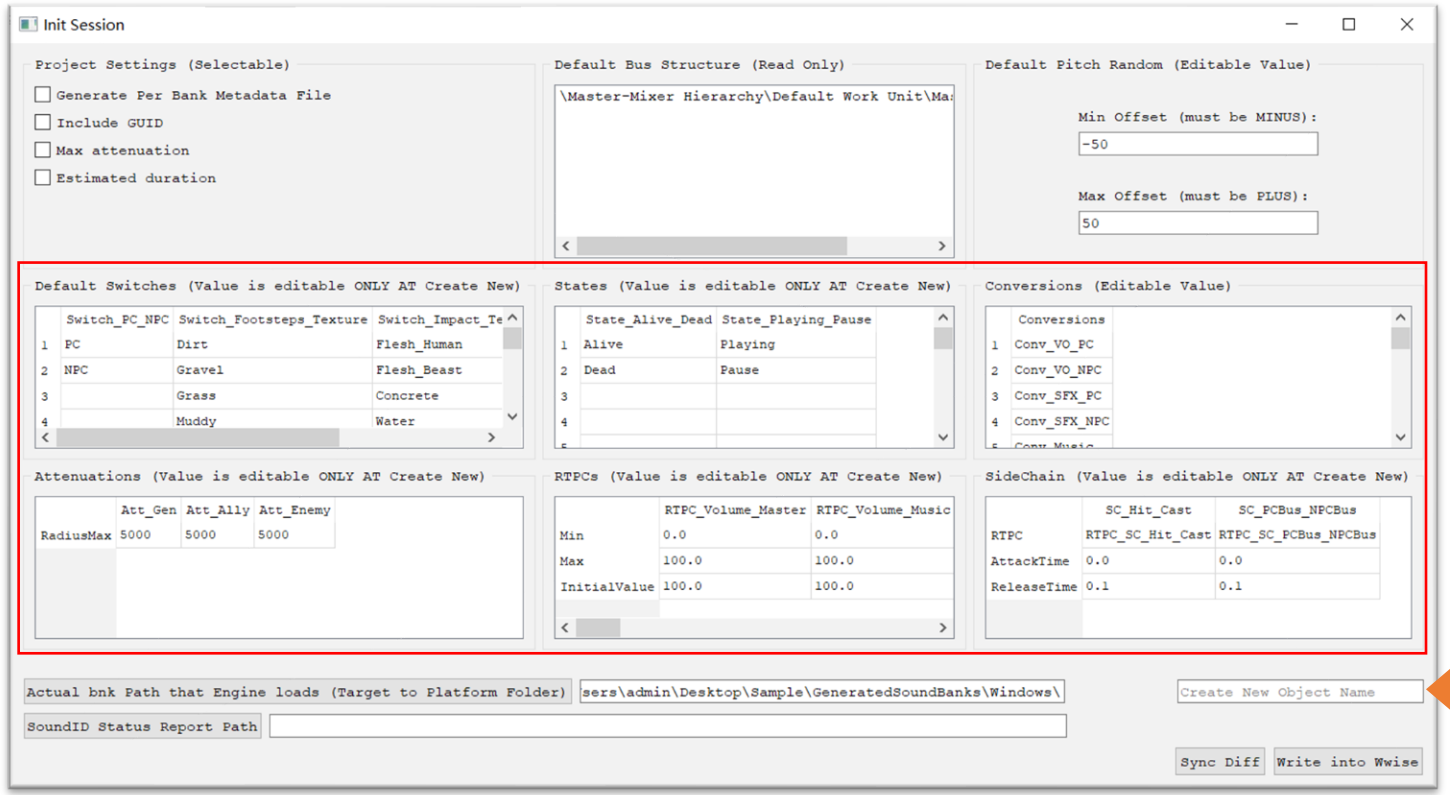
Project Initialization (First Time)

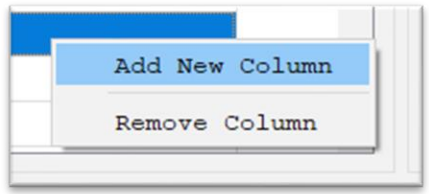
The remaining 6 areas correspond to the paths of Switches, States, Conversions, Attenuations, Game Parameters, and Effects (Meter) in the project, respectively.

If you need to add a column in a certain area, you need to first **enter the name of the object to be added in the text box in the bottom right corner**, then right-click in the target table and select "Add New Column".

If you need to remove a column in a certain area, directly right-click on any cell in the column to be deleted in that area and select "Remove Column".

Note: In the "Conversions" area, you can only add or delete text in cells, but cannot edit rows or columns.





Project Initialization (First Time)

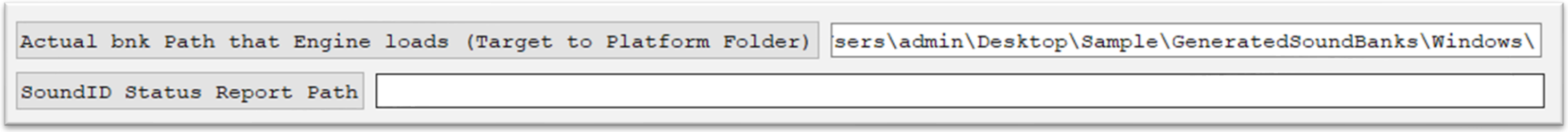
At the bottom, there are two more path information:

Actual bnk path that Engine loads:
In actual projects, the bnk files generated by Wwise usually need to be placed in a certain engine directory and eventually packaged together with other game data.
The manually assigned bnk path here is the actual path read by the engine, not the default GeneratedSoundBanks path under the Wwise project path.

Note: In the main interface, there is a right-click function called "Fill BankInfo", which is used to confirm and locate which bnk file an Event actually exists in. The query path for this function is the path here.

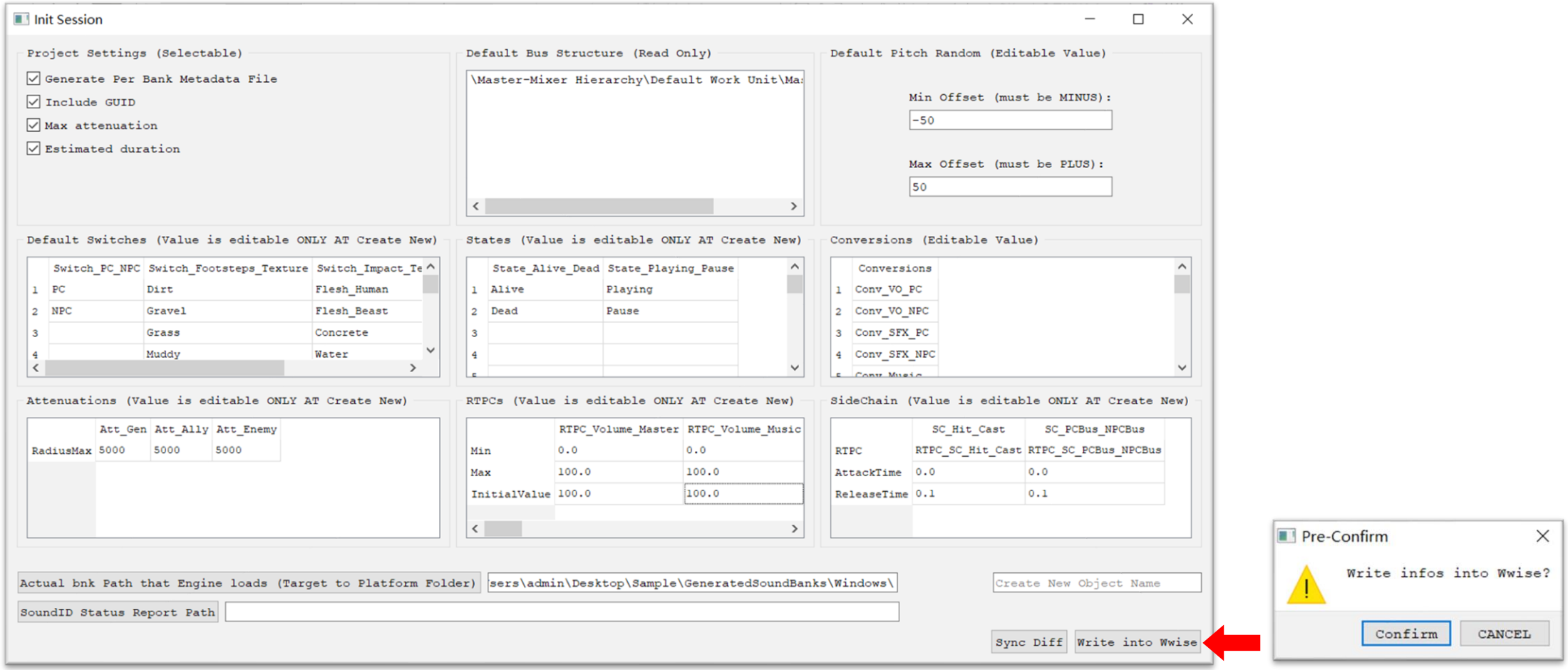
SoundID status Report Path:
There is a function module not included in this plugin that can compare all the sound IDs configured in the engine, skill editor, action editor, map editor, scripts, tables, etc., with all the sound IDs in the master audio table of this plugin, and organize the mismatched sound IDs and their locations into a JSON file for QA and as a reminder for designers.
The path here refers to the path of the comparison result JSON file exported by an additional function developed on the engine side.

Note: If such a function does not exist in your engine environment, this path location is not very useful and can be ignored. This is just an explanation, because the developer's own engine environment has it. ^_^



Project Initialization (First Time)

After editing the required content, click the "Write into Wwise" button in the bottom right corner. A confirmation window will pop up before execution. If there are no issues, click confirm to write all the content into Wwise.



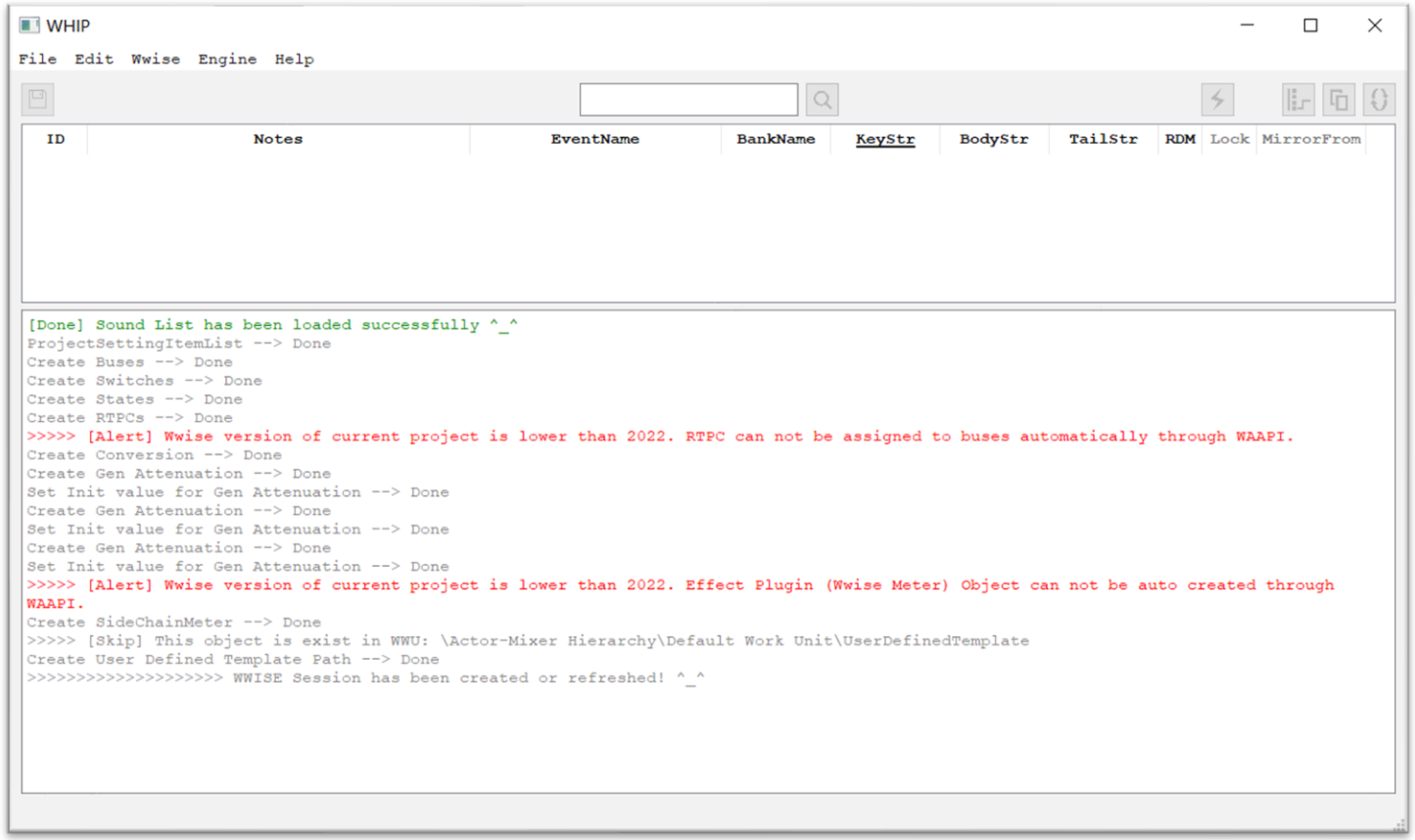
Project Initialization (First Time)

After writing is completed, the Init Session window will automatically close. We can check the execution log in the log area of the main interface and then check the results in Wwise.

Note: Since some WAAPI interfaces are only available in **versions higher than 2022**, some content cannot be written normally in lower versions, so you may see a few warning prompts similar to those in the figure below. Of course, it can also be done by directly modifying the wwu, but it hasn't been handled this way for now. But it's okay, for content that doesn't support automation, you can manually create it in Wwise later.

(The red text prompt about RTPC in the first line of the figure refers to the failure to assign the several RTPCs that control the volume output of the master bus, SFX, VO, and Music respectively to their corresponding buses after creating the buses.)

Reminder:
Please try to use the highest and relatively stable version of Wwise whenever possible.^_^

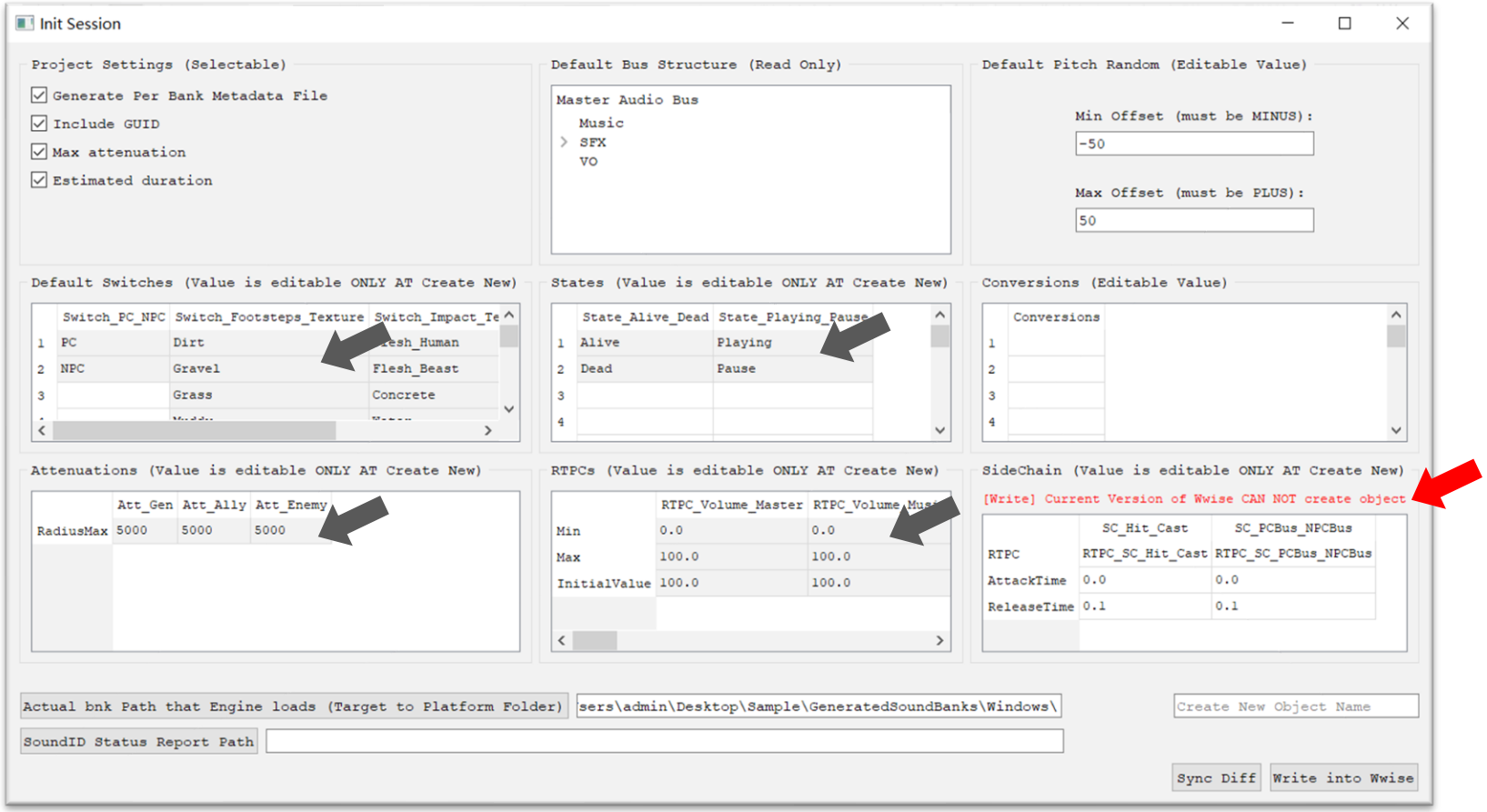


Project Initialization (NOT First Time)

If the Wwise project is not a clean new project and already has a lot of content, when we open the Init Session window and click the "Sync Diff" button, we will see the existing content in Wwise displayed in a gray background and non-editable state.

For safety reasons, the existing information is not allowed to be modified a second time through the plugin, and can only be manually changed in Wwise. However, we can continue to add content that doesn't exist in Wwise through the plugin window.

Note: You can also see some reminders about unsupported features. Those that are not supported for reading or writing will not take effect, so there's no need to spend time on them.

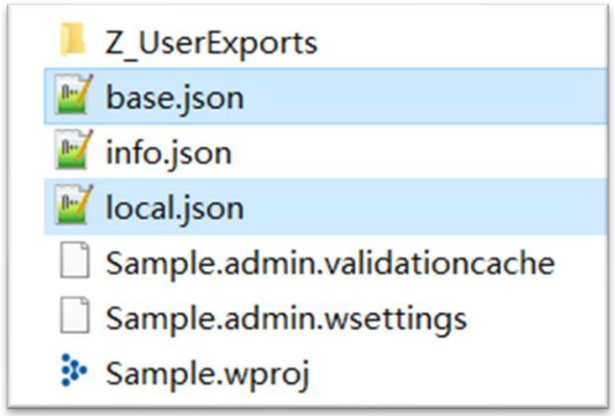


Project Initialization (NOT First Time)

Reminder:
Some of the data in the Init Session window will be recorded in the **base.json** file in the current Wwise project directory, while another part will be recorded in the **local.json** file.

The text changes in the Init Session window will be automatically saved after input is completed, without the need for manual saving.

Note: For data security and plugin stability, please modify the content through the plugin window and try not to manually change the JSON file directly!



base.json

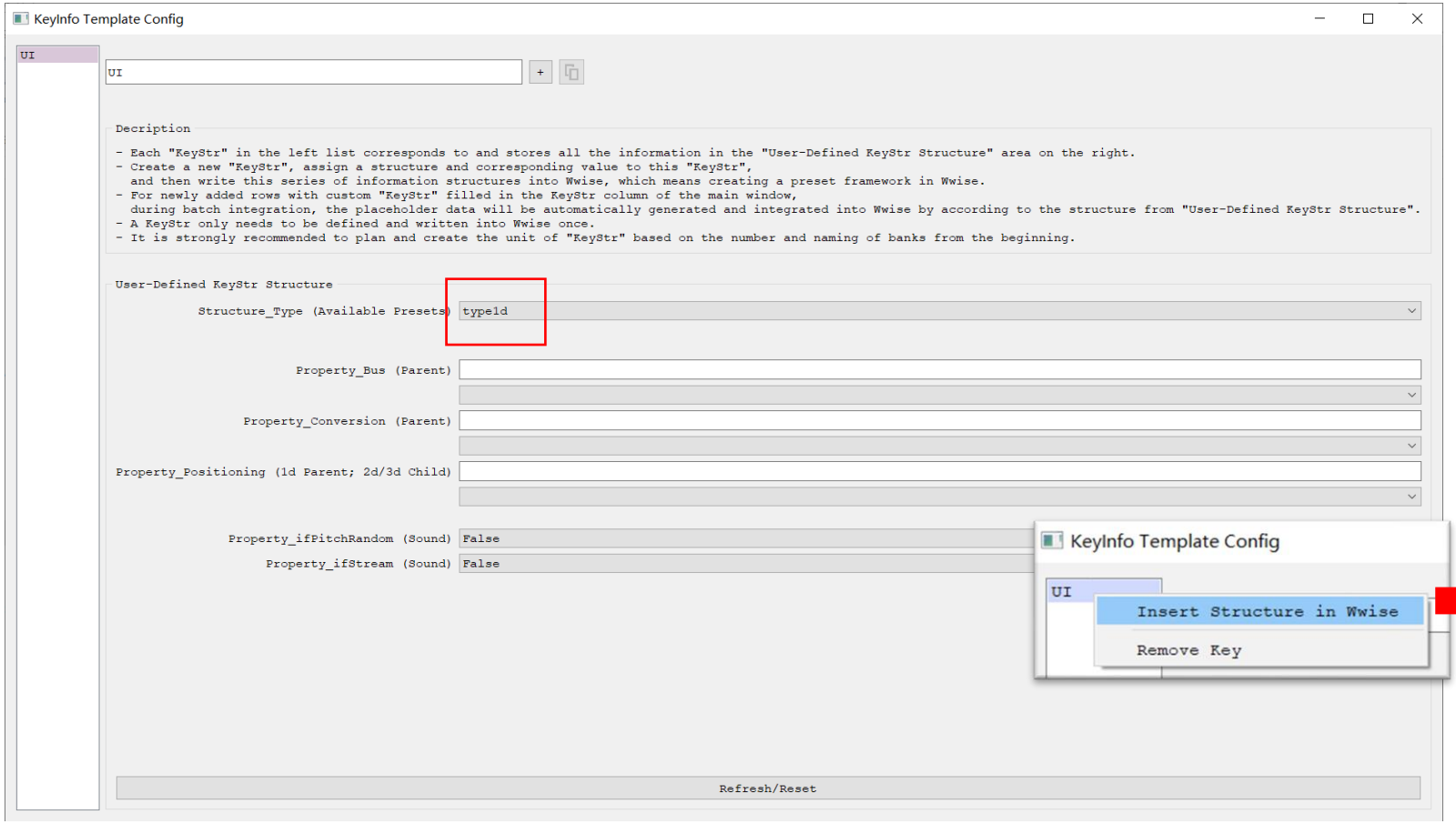

```
"Init_ProjectSettings": {
    "GenerateMultipleBanks": "True",
    "SoundBankGeneratePrintGUID": "True",
    "SoundBankGenerateMaxAttenuationInfo": "True",
    "SoundBankGenerateEstimatedDuration": "True"
},
"Init_BUS": {
    "Master Audio Bus": {
        "SFX": {
            "Action": {
                "PC": "",
                "NPC": ""
            },
            "Amb": "",
            "UI": ""
        },
        "Music": "",
        "VO": "",
        "CG": ""
    }
},
"Init_Switch": {
    "Switch_PC_NPC": [
        "PC",
        "NPC"
    ],
    "Switch_Footsteps_Texture": [
        "Dirt",
```


local.json

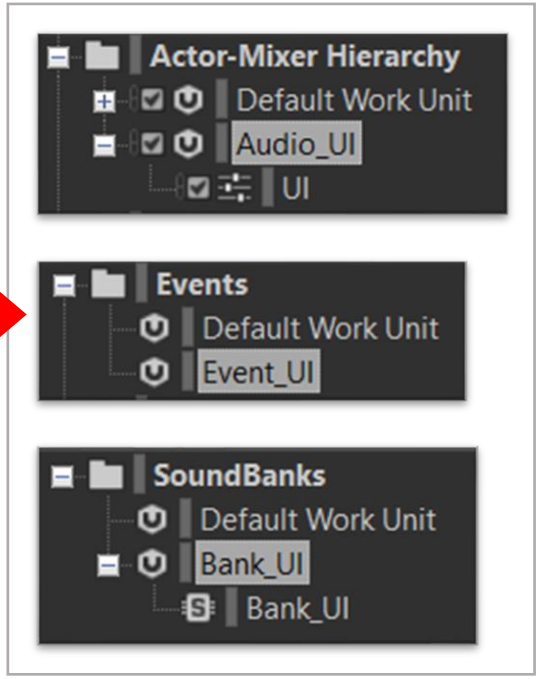

```
"$ProjectStr$": "Sample.wproj",
"-------------[Local Path Info]-------------": "----------------------",
"ActualGeneratedSoundBankPathOfOnePlatform": "C:\\Users\\admin\\Desktop\\Sam
"Path_SoundIDSTatusReport": "C:\\Program Files (x86)\\Audiokinetic\\Wwise 20
"Path_DefaultSaveAsFolder": "C:\\Users\\admin\\Desktop"
```

How to integrate a simple UI sound effect (type1d)

Step 1: Through the "KeyInfo Template Config" window, add a new KeyStr (as keyword), such as: UI. After setting a series of attributes for the UI KeyStr, right-click the KeyStr and automatically write the KeyStr structure into Wwise.



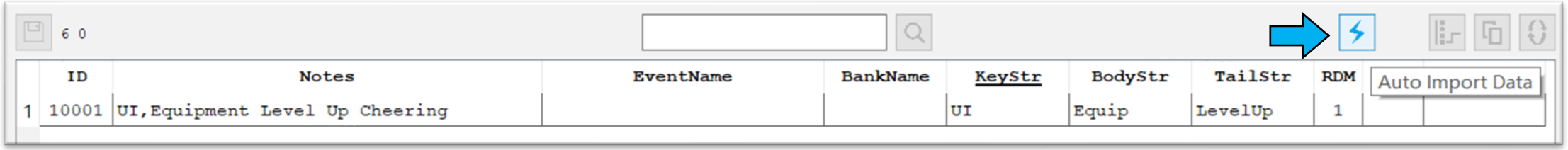
For detailed operating instructions on creating new KeyStr and attributes, please read the "Set it and forget it! Configure Automation Templates!" section.



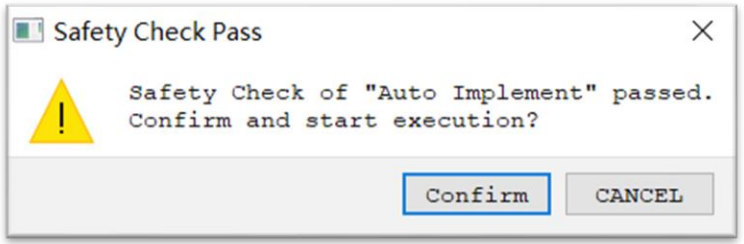
How to integrate a simple UI sound effect (type1d)

Step 2: In the main interface, add a new row. Fill in "UI" in the KeyStr column as the beginning of the naming. Then fill in the other parts of the naming, the "RDM" random number, and the "Notes" comment.

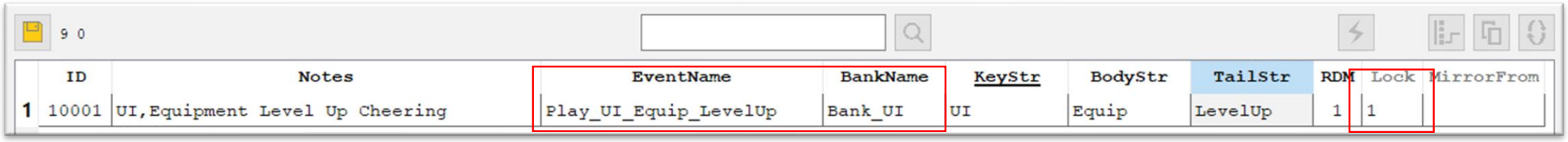
Step 3: After saving, click the "Auto Import Data" button. The safety pre-check window will pop up. If the prompt passes, click confirm and wait for the automated integration to complete. If the prompt does not pass, fix the problem first and then re-execute.



	ID	Notes	EventName	BankName	KeyStr	BodyStr	TailStr	RDM
1	10001	UI,Equipment Level Up Cheering			UI	Equip	LevelUp	1



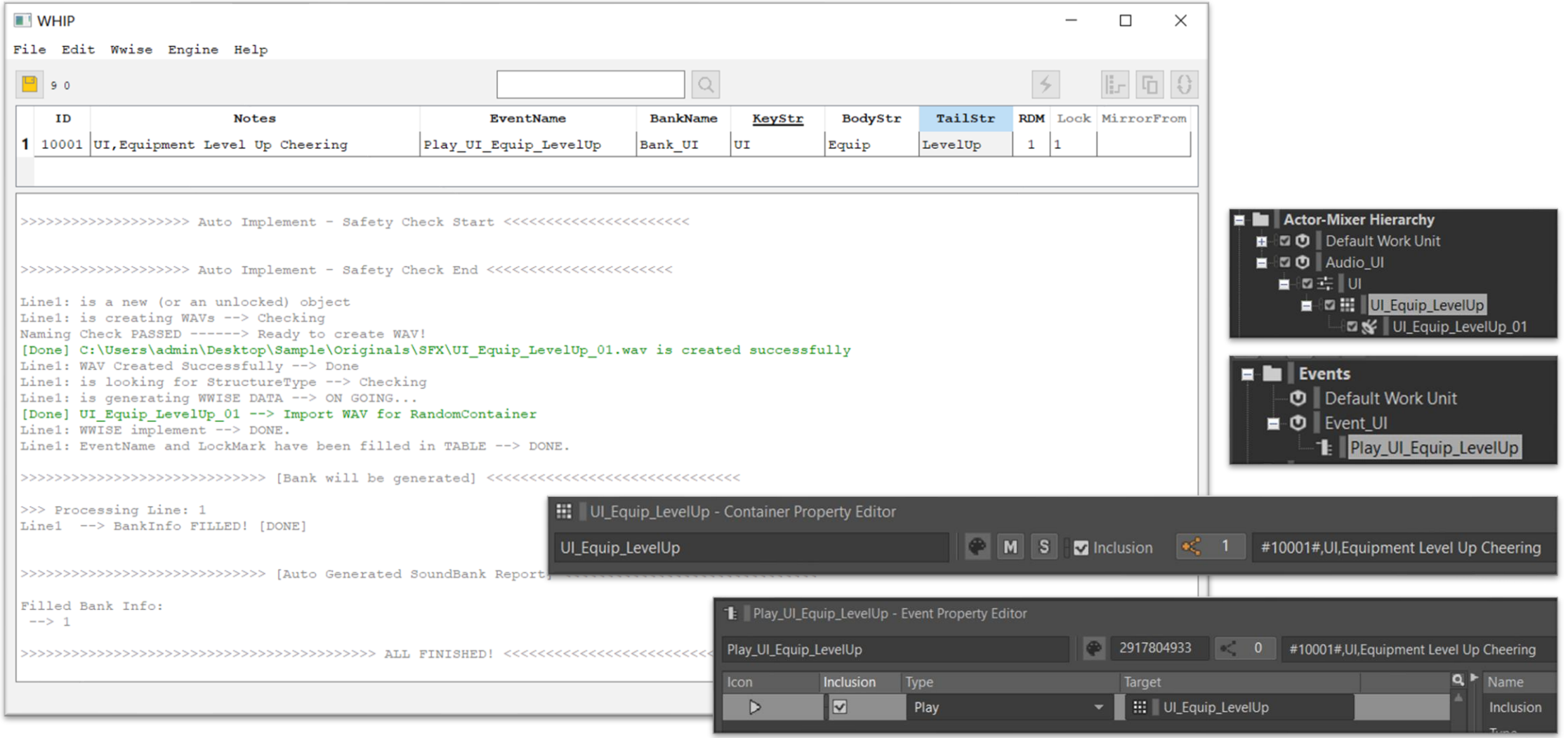
Step 4: After execution, pay attention to the changes in the EventName, BankName, and Lock columns in the table. Open Wwise to confirm the integration results. Save the table data, replace the temporary silent placeholders with the official resources, and the process ends.



	ID	Notes	EventName	BankName	KeyStr	BodyStr	TailStr	RDM	Lock	MirrorFrom
1	10001	UI,Equipment Level Up Cheering	Play_UI_Equip_LevelUp	Bank_UI	UI	Equip	LevelUp	1	1	

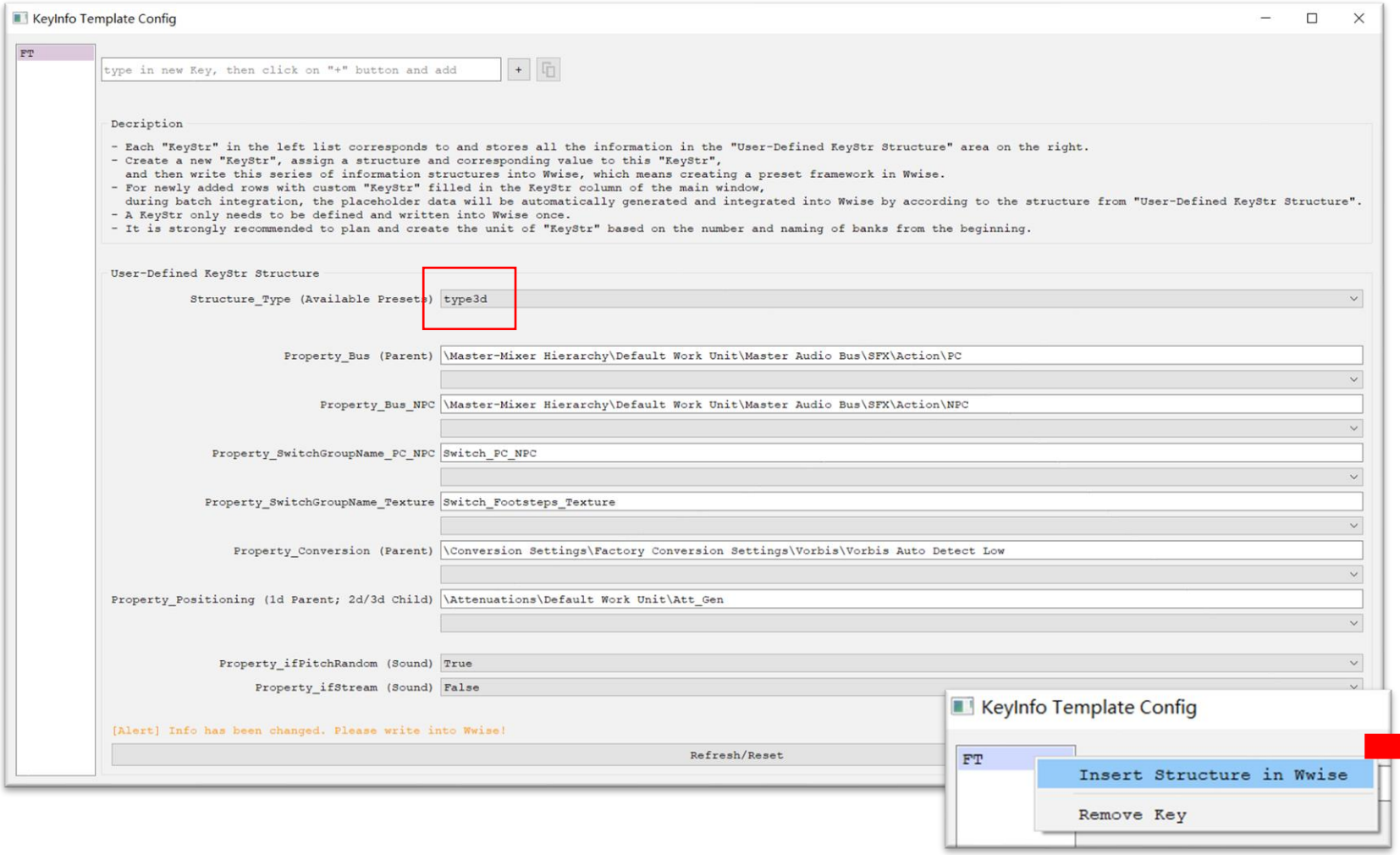
How to integrate a simple UI sound effect (type1d)

View results:

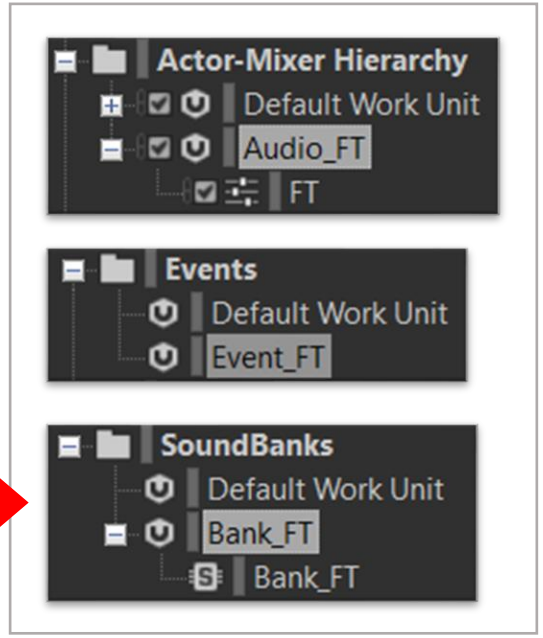


How to integrate a set of player footsteps (type3d)

Step 1: Through the "KeyInfo Template Config" window, add a new KeyStr (as keyword), such as: FT. After setting a series of attributes for the FT KeyStr, right-click the KeyStr and automatically write the KeyStr structure into Wwise.



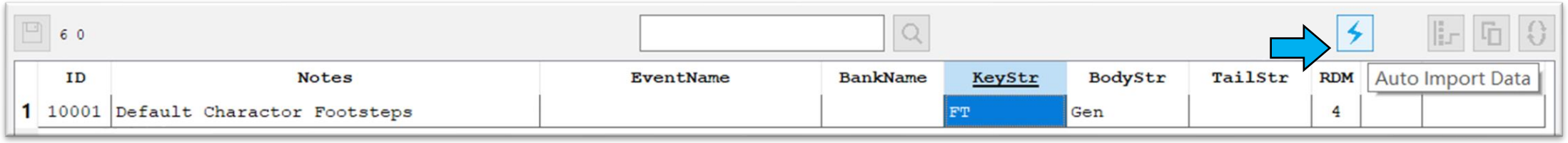
For detailed operating instructions on creating new KeyStr and attributes, please read the "Set it and forget it! Configure Automation Templates!" section.

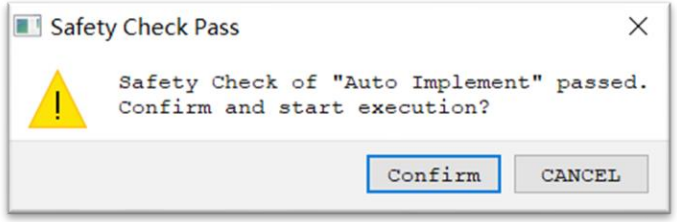


How to integrate a set of player footsteps (type3d)

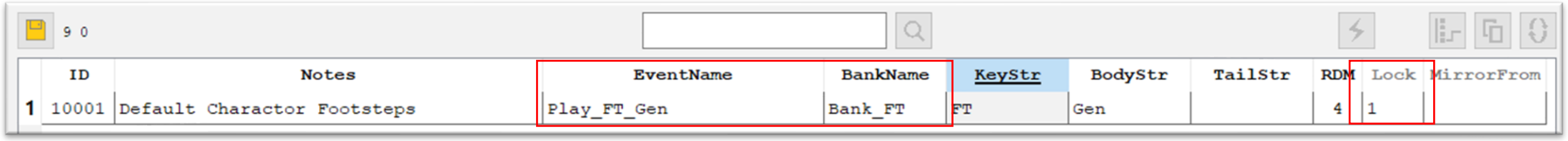
Step 2: In the main interface, add a new row. Fill in the "FT" KeyStr in the KeyStr column as the beginning of the naming. Then fill in the other parts of the naming, the "RDM" random number, and the "Notes" comment.

Step 3: After saving, click the "Auto Import Data" button. The safety pre-check window will pop up. If the prompt passes, click confirm and wait for the automated integration to complete. If the prompt does not pass, fix the problem first and then re-execute.



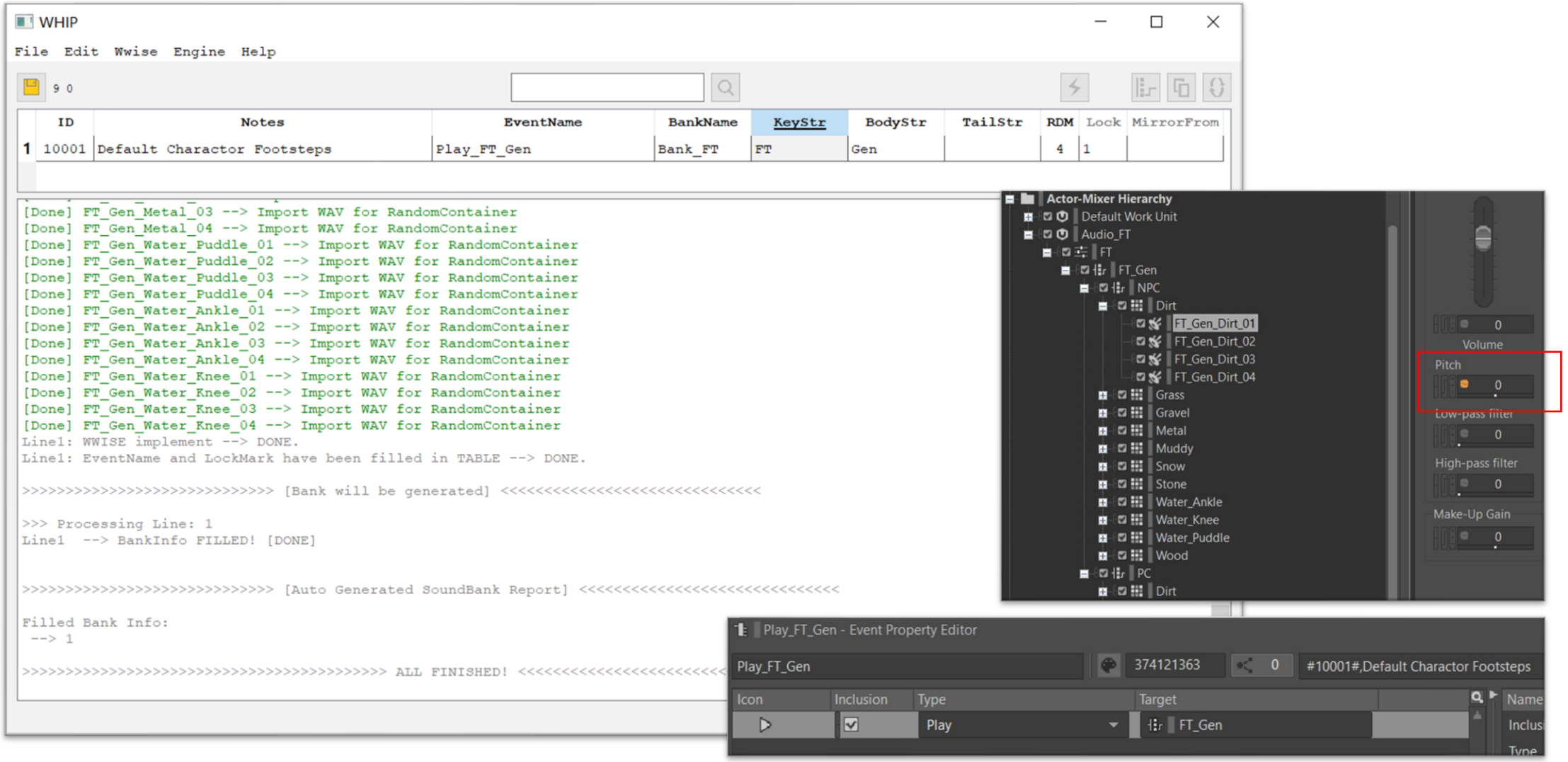


Step 4: After execution, pay attention to the changes in the EventName, BankName, and Lock columns in the table. Open Wwise to confirm the integration results. Save the table data, replace the temporary silent placeholders with the official resources, and the process ends.



How to integrate a set of player footsteps (type3d)

View results:



How to integrate a set of player footsteps (type3d)

【Supplementary Explanation】

For data generated through type3d (including type2d and type2d_vo), the wav resources in the containers on both the PC and NPC sides share the same set of resources, and there is no duplication at the resource level. It's not two separate sets for PC and NPC, but the same set of wav resources assigned to both PC and NPC.

However, sometimes designers don't want PC and NPC to share the same set of resources, but rather have them differentiated. In that case, the typet (user-defined type) template type would be more suitable than the type3d template. For specific configuration methods of typet, please refer to the typet example.

***Extended Tip:**

Between type1d and type3d, there is another template type called type2d.

Compared to type3d, the attribute structure of type2d only retains the Switch_PC_NPC hierarchy suitable for multi-person signal switching and reduces the Texture hierarchy suitable for material switching.

Therefore, if a designer wants to create a sound that does not involve material switching, such as a Whoosh sound of a player swinging a torch, they can choose type2d when configuring the KeyStr attributes.

***Reminder:**

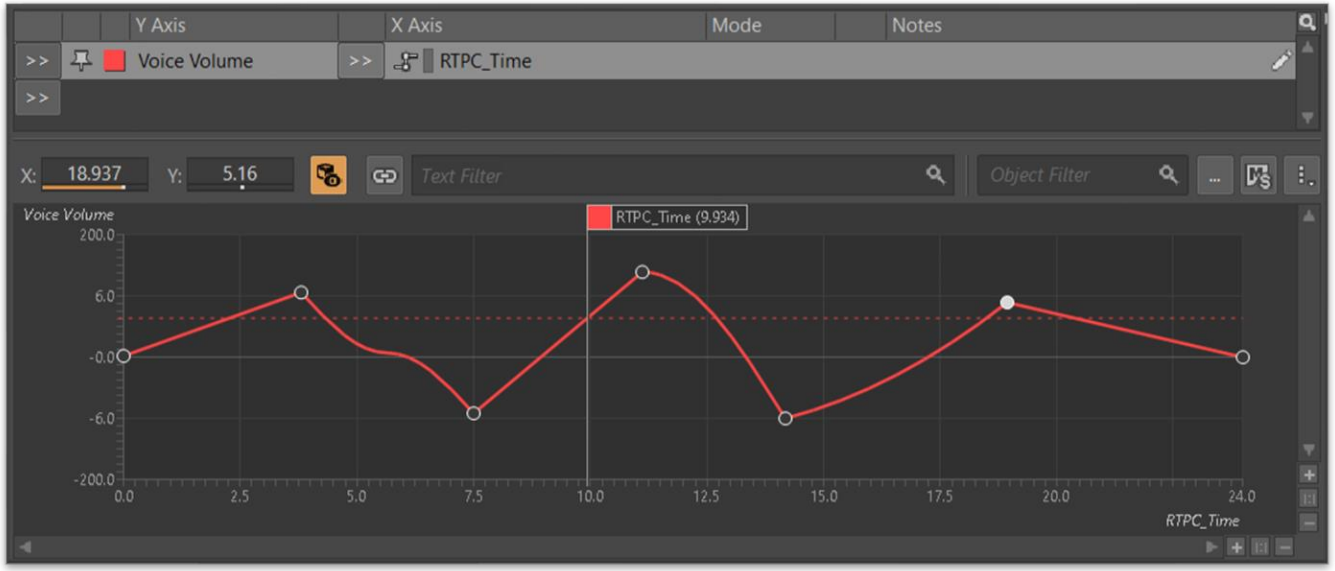
The type3d template structure is suitable for the integration and packaging of resources involving multiple character types and material switching. For example, basic movement sound effects where PC and NPC signals need to be managed independently and footstep materials involve switching; or collision sound effects involving weapon hits on different materials and attack signal paths that require differentiated management.

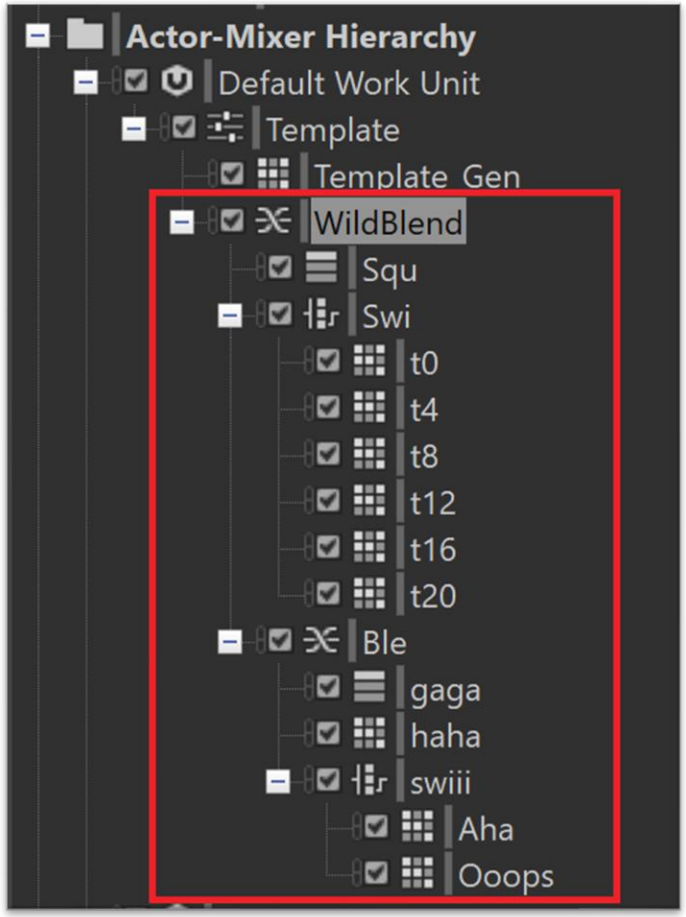
type3d is a template type with a "dual-layer Switch dimension" suitable for managing "PC/NPC signal differences + other Switch changes". Similarly, type2d is a template type with a "single-layer Switch dimension" suitable for managing "PC/NPC signal differences".

I have a custom template structure that is widely used, but its structure is quite special, and the plugin's three preset basic template frameworks cannot handle it. What should I do? (typet)

First, the designer needs to manually customize the template structure form within the Wwise project, for example, like the one shown in the illustration. To better explain and illustrate, the example template structure is intentionally made relatively complex to clearly showcase the details of the integrated results.

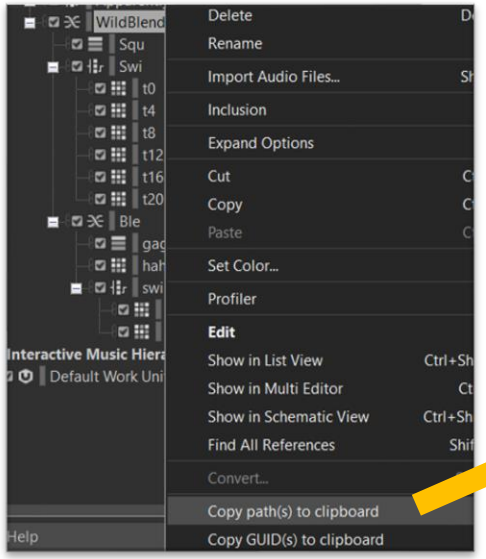
***Friendly reminder:** The "WildBlend" object in the illustration is purely an example (assuming the designer happened to design such a form) and has no real guiding significance. For the sake of your and your team's physical and mental health, please try to avoid creating such "wildly passionate" template structures in actual work.

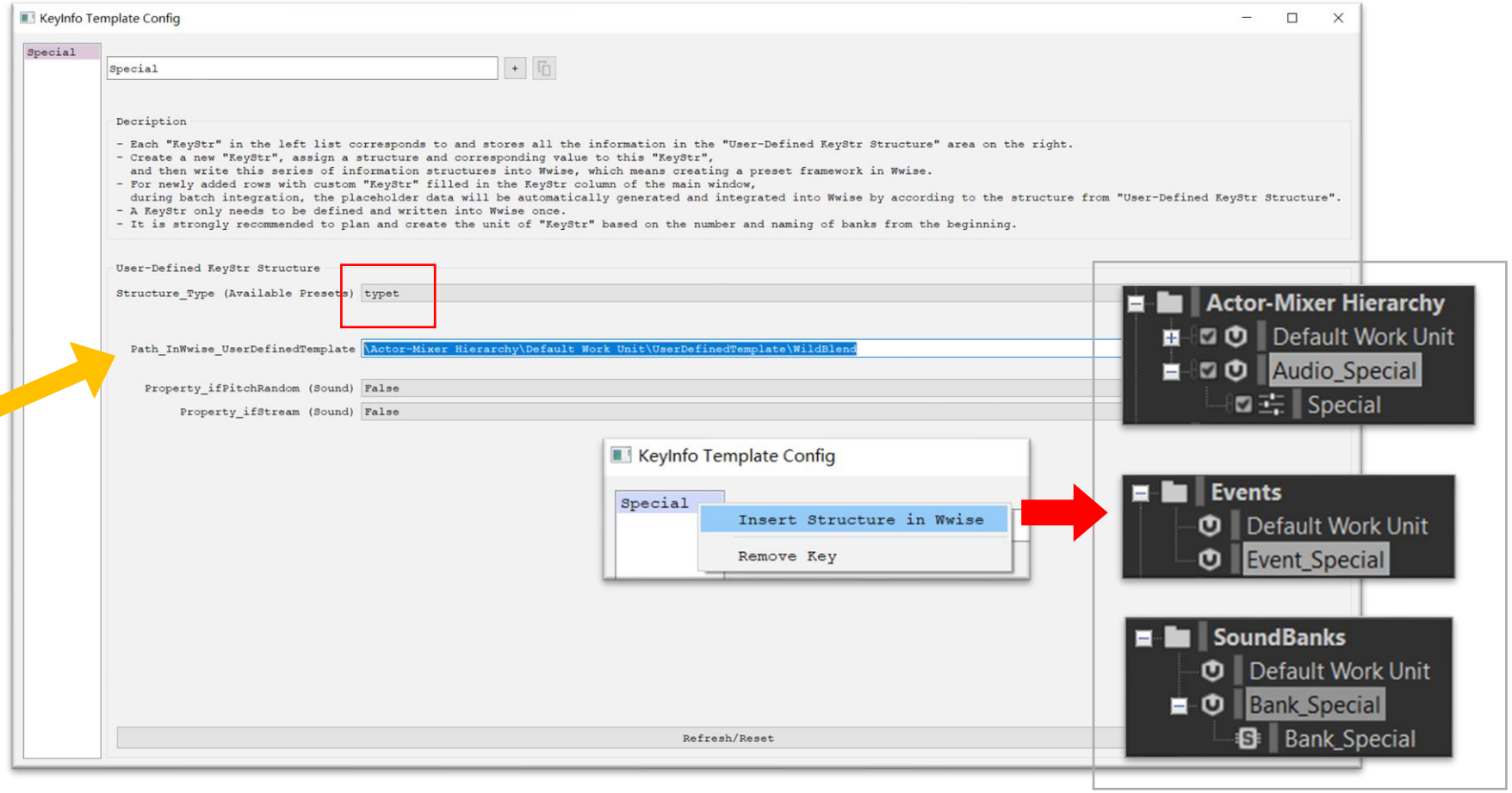




I have a custom template structure that is widely used, but its structure is quite special, and the plugin's three preset basic template frameworks cannot handle it. What should I do? (typet)

Step 1: Through the "KeyInfo Template Config" window, add a new keyword, such as: Special.
Step 2: Copy the template path from the Wwise project and paste it into the "Path_InWwise_IserDefinedTemplate". After confirming everything else is correct, right-click the KeyStr and automatically write the KeyStr structure into Wwise.



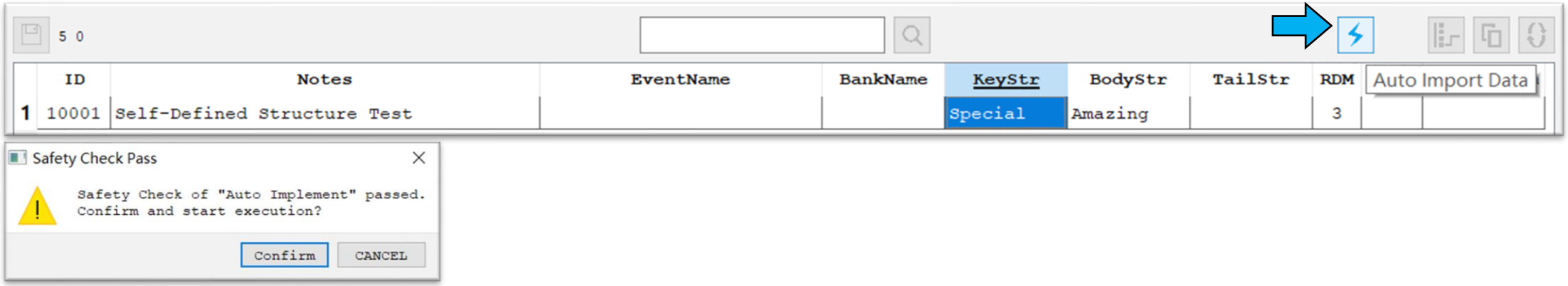


For detailed operating instructions on creating new KeyStr and attributes, please read the "Set it and forget it! Configure Automation Templates!" section.

I have a custom template structure that is widely used, but its structure is quite special, and the plugin's three preset basic template frameworks cannot handle it. What should I do? (typet)

Step 3: In the main interface, add a new row. Fill in the "Special" KeyStr in the KeyStr column as the beginning of the naming. Then fill in the other parts of the naming, the "RDM" random number, and the "Notes" comment.

Step 4: After saving, click the "Auto Import Data" button. The safety pre-check window will pop up. If the prompt passes, click confirm and wait for the automated integration to complete. If the prompt does not pass, fix the problem first and then re-execute.



	ID	Notes	EventName	BankName	KeyStr	BodyStr	TailStr	RDM
1	10001	Self-Defined Structure Test			Special	Amazing		3

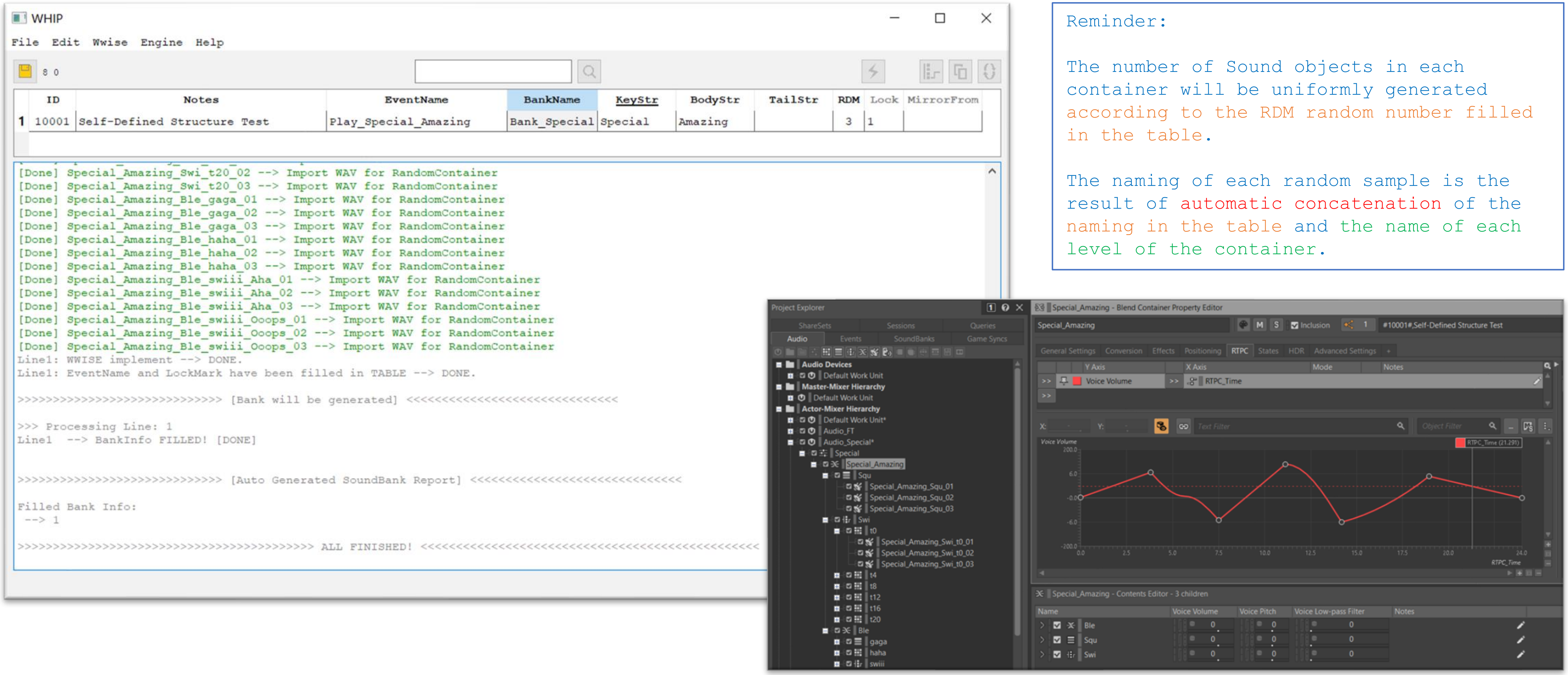
Step 5: After execution, pay attention to the changes in the EventName, BankName, and Lock columns in the table. Open Wwise to confirm the integration results. Save the table data, replace the temporary silent placeholders with the official resources, and the process ends.

	ID	Notes	EventName	BankName	KeyStr	BodyStr	TailStr	RDM	Lock	MirrorFrom
1	10001	Self-Defined Structure Test	Play_Special_Amazing	Bank_Special	Special	Amazing		3	1	

Reminder:
The difference between the data generated by typet and type2d/3d is that each wav is an independent resource, and there is no resource reuse at the resource level.

I have a custom template structure that is widely used, but its structure is quite special, and the plugin's three preset basic template frameworks cannot handle it. What should I do? (typet)

View results:



Reminder:

The number of Sound objects in each container will be uniformly generated according to the RDM random number filled in the table.

The naming of each random sample is the result of automatic concatenation of the naming in the table and the name of each level of the container.

I have a custom template structure that is widely used, but its structure is quite special, and the plugin's three preset basic template frameworks cannot handle it. What should I do? (typet)

【Supplementary Explanation】

For version 2.0 of the plugin, considering the data security, easy maintenance, and easy extension of one-click batch automation, the developer has established the following rules:

1) Since version 2.0 of the plugin does not yet support dynamic voice and interactive music modules, the following container types are temporarily listed as **DISABLED**:

Voice type; **Music Switch Container** type; **Music Playlist Container** type; **Music Segment** type

2) The container types that can be freely combined and used are: **Random Container** type; **Sequence Container** type; **Blend Container** type; **Switch Container** type. The **DISABLED** container types are: **Sound** type; **Actor-Mixer** type; **Virtual Folder** type.

*Since the involvement of these three(Sound, Actor-Mixer, Virtual Folder) types can easily bring many unnecessary troubles to automation and subsequent maintenance, version 2.0 temporarily disables these types. After the developer finds an appropriate solution, subsequent versions will consider recalling them one by one.

3) In a legal custom container nesting structure, the last-level container is not allowed to end with a lone Switch Container.

*This is simply to remind designers not to forget to arrange corresponding content assignments for the Switch container when making templates. However, there is currently no such safety check for the Switch containers at earlier levels. If the designer forgets, they really forget, and some lone Sounds will naturally be generated in the Switch container after automation.

4) It is allowed, but not recommended, to have a structure where "**three consecutive levels** are all Switch Containers".

*Note: Due to data security and complexity considerations, the "One-click Expand Switch" function does not currently support handling this structure.

Naming Conventions

Global naming character rules: Except for English uppercase and lowercase letters, numbers, and underscores, all other symbols are illegal characters that are not allowed in naming!

First, when we want to add a new object in the main interface table, we need to add a row in the main interface first.
You can add a row by selecting "Edit→Add Line" in the menu bar or by using a shortcut key.

Then, in the newly added row, there are several cells that require special attention:

1) In the ID cell, enter an ID value (required. Each time a new row is added, the program will automatically fill in a sequential value, which usually does not need to be modified unless there are special requirements).
Tip: ID is a required field! It can be a number, a letter, or a combination of numbers and letters. One ID corresponds to one row of data, and each ID must be globally unique and cannot be duplicated.

2) In the KeyStr cell, enter a value (required. It must be a KeyStr that has been successfully created through the "KeyInfo Template Config" window and successfully written into Wwise).
Tip: Except for the strings created by the designer, all other strings will be considered as illegal characters.

3) In the BodyStr and TailStr cells, fill in the required naming fields.
Tip: Both the BodyStr and TailStr cells can be filled, or you can choose to fill one of them. If both cells have information, the final naming will be concatenated using an underscore.

4) In the RDM cell, enter the required random number (required).
Tip: The legal range for the random number is an integer from 1 to 8.

5) In the Notes cell, enter the required comment information (not affected by the "Global Naming Character Rules").
Tip: After one-click integration, the comment information will be automatically synchronized to the Notes of the corresponding Event in the WWISE project.

6) The contents of the EventName and BankName cells will be automatically filled in by the plugin function after automatic integration, and do not need to be filled in manually.

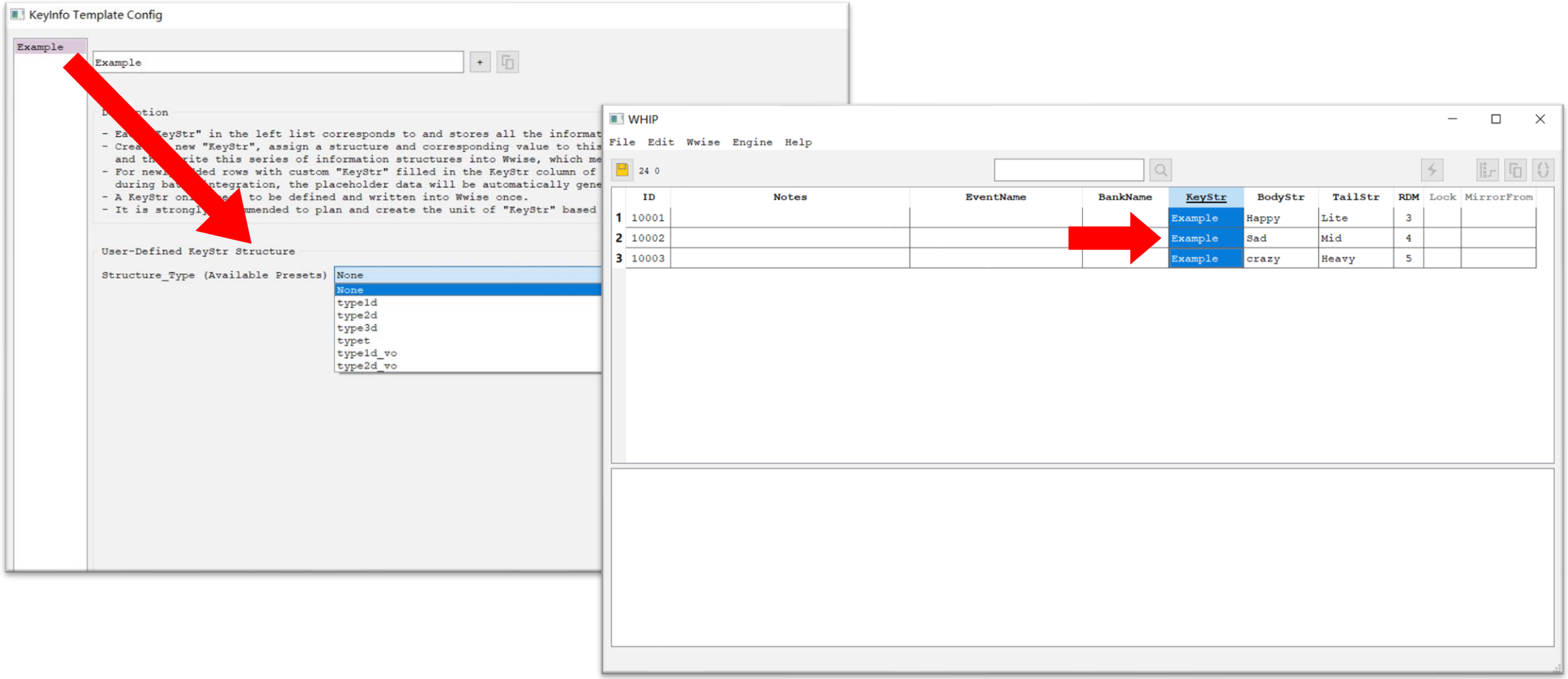
7) Rows that have undergone automatic integration will automatically have a character written in the Lock cell, such as "1", to serve as an "execution mark". If the user accidentally includes a "row with an execution mark" while processing other rows, the program will automatically skip that row during execution and provide a prompt in the log.

Note: If you need to process the data in that row again through the automation program later, you need to clear both the **Lock** and **BankName** cells first.

Set It and Forget It! Configure Automation Templates!

In the process of achieving "naming is everything", there is one step that is particularly important, which is to use a **KeyStr** as a bridge to associate **the object name containing the keyword** with **the preset attributes**.

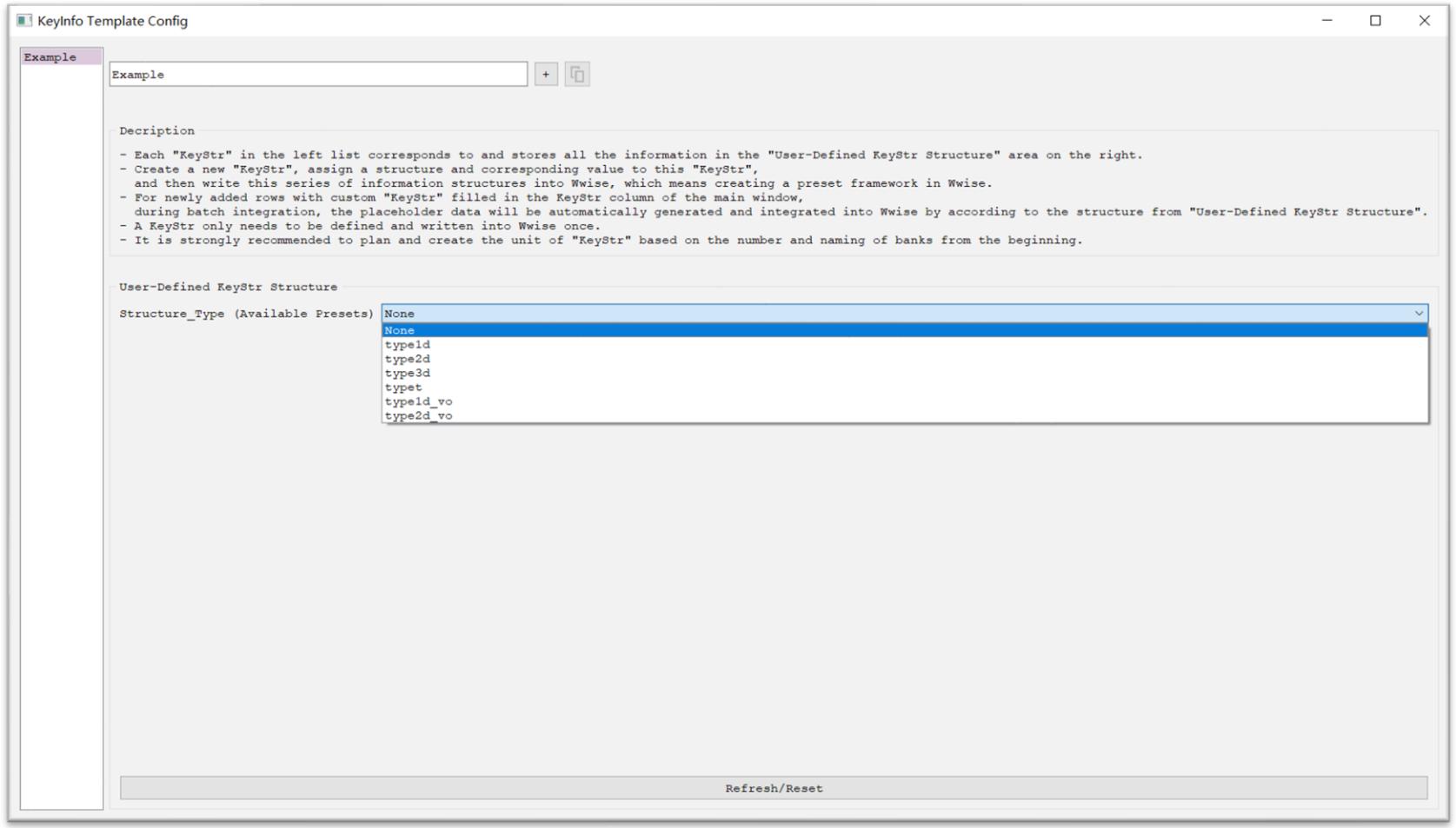
Principle: When we create a new KeyStr "Example" and assign it a series of custom attributes, and write the attribute framework into Wwise, any object with the KeyStr "Example" at the beginning of its name will generate data in the corresponding WorkUnit in Wwise according to the attribute settings in its "User-Defined KeyStr structure" during one-click execution.



Set It and Forget It! Configure Automation Templates!

First, enter a **KeyStr** in the text box and click the "+" button on the right. At this point, the new KeyStr will appear in the list on the left, with **a pink background color**.

The first attribute that needs to be set for the KeyStr is "Structure_Type (Available Preset)". Different choices correspond to different **container structure forms**.

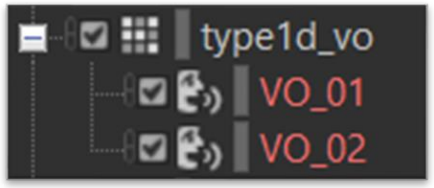


Set It and Forget It! Configure Automation Templates!

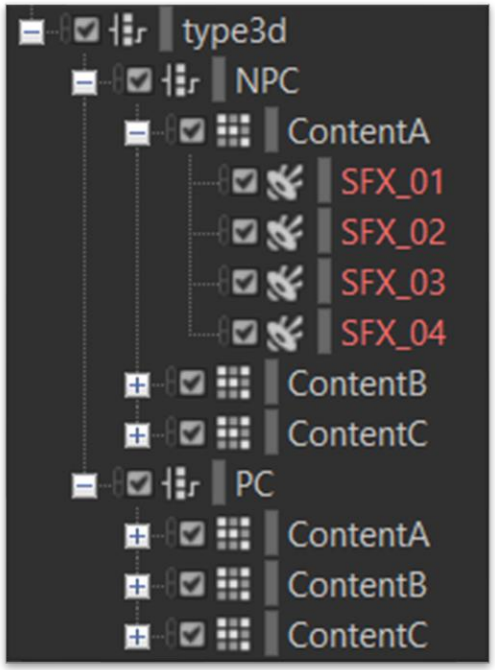
Example: The following shows the different **container structure forms** corresponding to each template structure type.
Except for typet, which can be customized by the designer, the other container structures are fixed and ready-to-use forms.

type1d:
Single Random Container

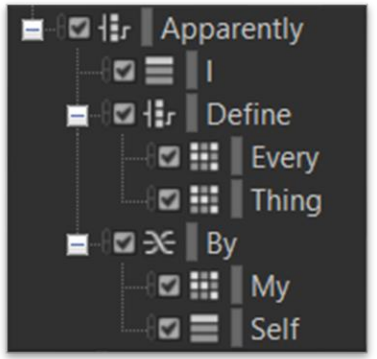
type1d_vo:
Single Random Container(for voice)



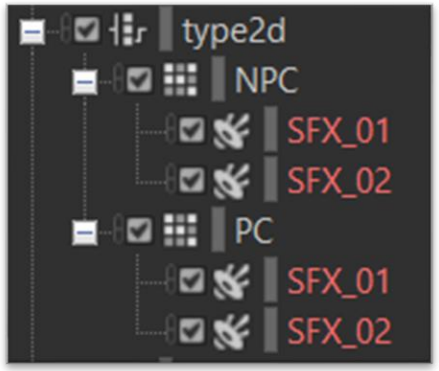
type3d:
Dual-Layer Switch Container



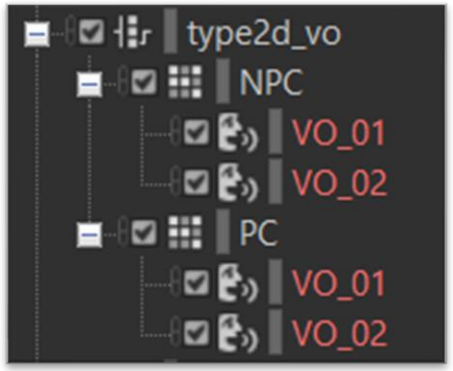
typet:
Self-Defined



type2d:
Single-Layer Switch Container



type2d_vo :
Single-Layer Switch Container(for voice)



Note: The above structure types might can basically cover the vast majority of game types and most of practical needs of objects.

Set It and Forget It! Configure Automation Templates!

Example: Attributes of type1d and type1d_vo

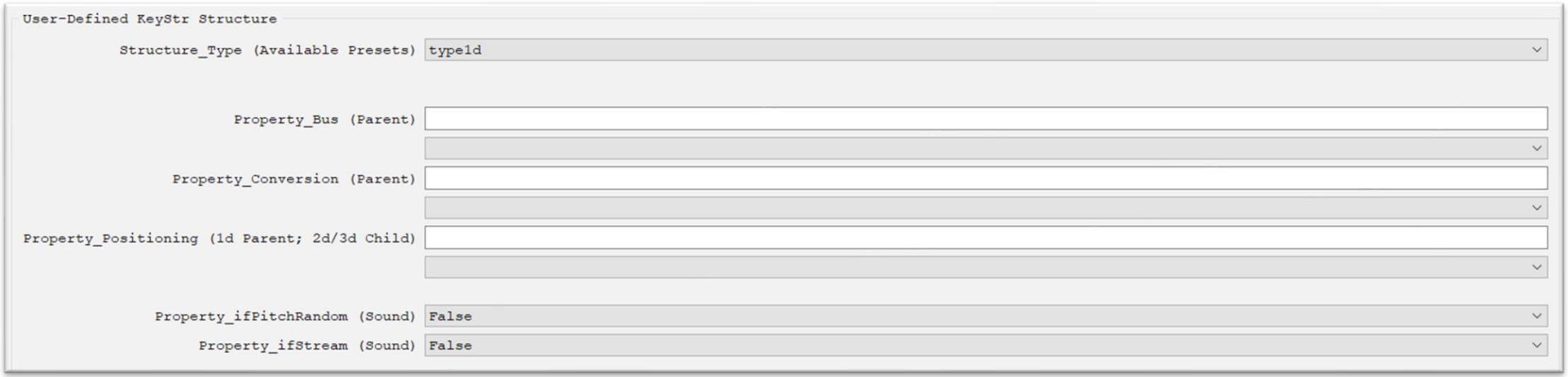
Property_BUS (Parent): Here, you select the Bus on the top-level Actor-Mixer generated by the current KeyStr. This way, the Bus of all child objects will default to follow the parent.

Property_Conversion (Parent): Same as above, it also selects and assigns a Conversion to the top-level Actor-Mixer.

Property_Positioning (1d Parent; 2d/3d Child): Since the current template structure type is a single Random container type of 1d, you can select an Attenuation object and directly assign it to the top-level Actor-Mixer.

Property_ifPitchRandom (Sound): If "False" is selected, the batch-generated Sound objects will not apply the custom random attribute values in the "Init Session" window; otherwise, they will be automatically applied.

Property_ifStream (Sound)": If "False" is selected, the batch-generated Sound objects will not automatically check "Stream"; otherwise, they will be automatically checked.



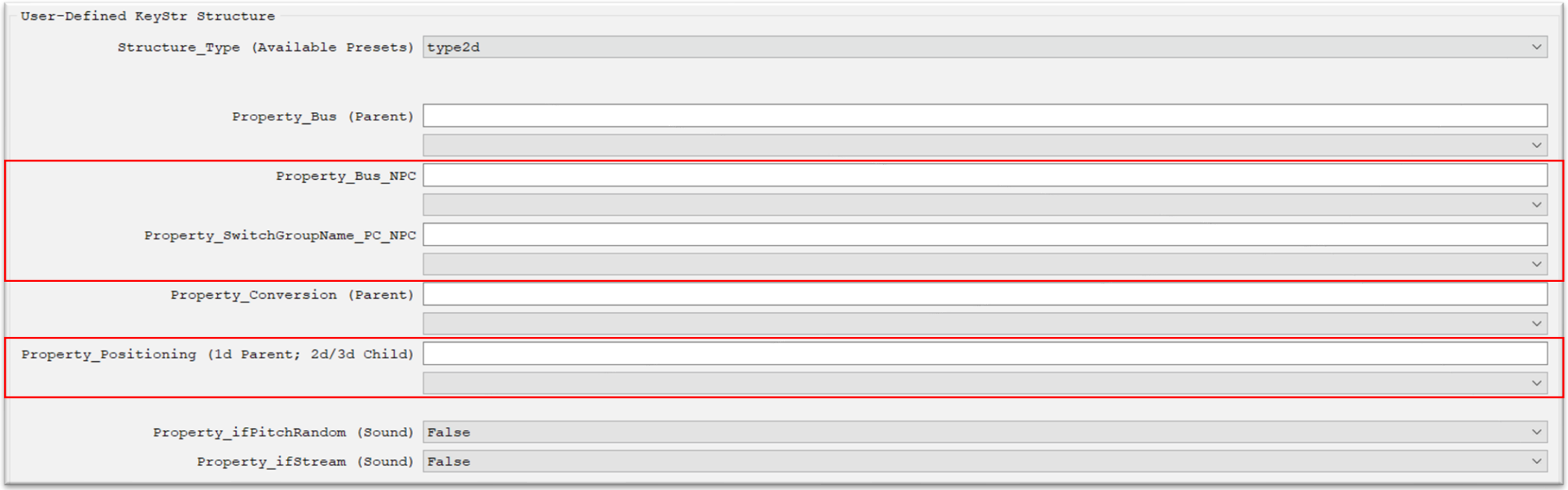
Set It and Forget It! Configure Automation Templates!

Example: There are two attributes in type2d and type2d_vo that are different from 1d type

Property_Bus_NPC: Here, you select a BUS that manages NPC signals and assign it to the random container corresponding to the NPC data under each newly added Switch container.

Property_SwitchGroupName_PC_NPC: Assign a SwitchGroup responsible for managing PC/NPC to each newly added Switch container.

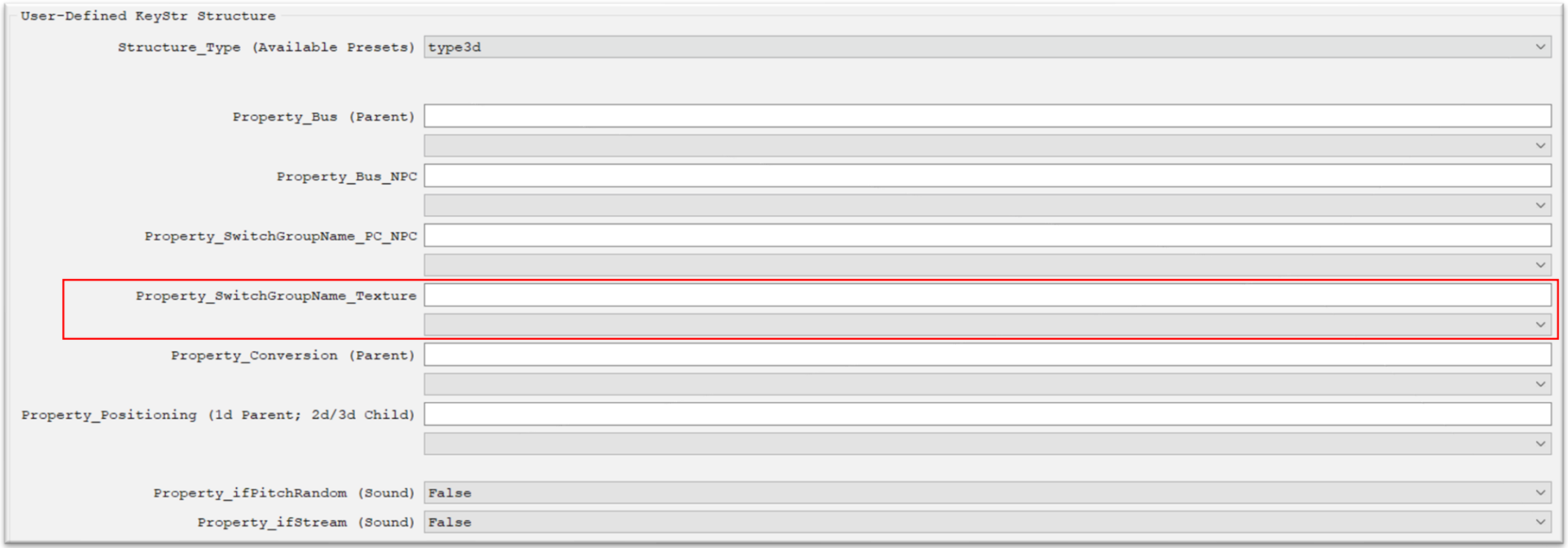
Property_Positioning (1d Parent; 2d/3d Child): Since the current type is 2d, the Attenuation will be assigned to the random container corresponding to the NPC data under the Switch container, rather than the top-level Actor-Mixer container.



Set It and Forget It! Configure Automation Templates!

Example: type3d has another necessary attribute that is different from 1d and 2d types

Property_SwitchGroupName_Texture:
Here, you select a SwitchGroup corresponding to the material and assign it to the Switch container corresponding to PC and NPC.



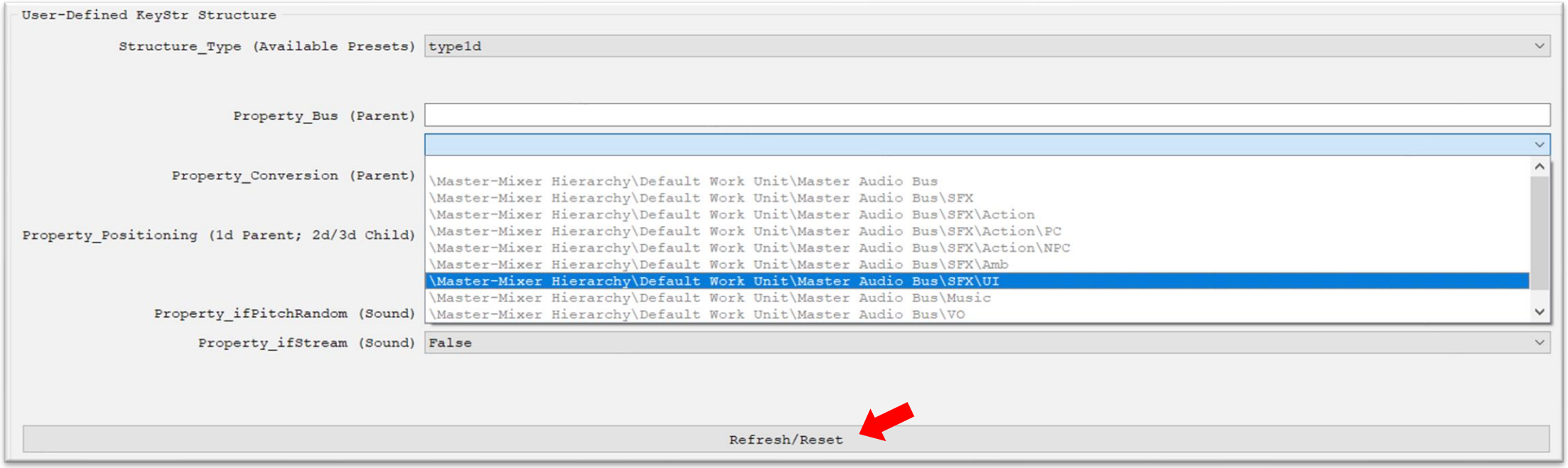
Set It and Forget It! Configure Automation Templates!

Filling Example: Attributes of type1d, type1d_vo, type2d, type2d_vo, type3d

Below each text box, there is a **drop-down list** showing the currently available legal object paths in the project.

After clicking, the path or object name information will be directly displayed in the text box, **without the need for manual input**.
Of course, you can also directly fill in or modify it manually.

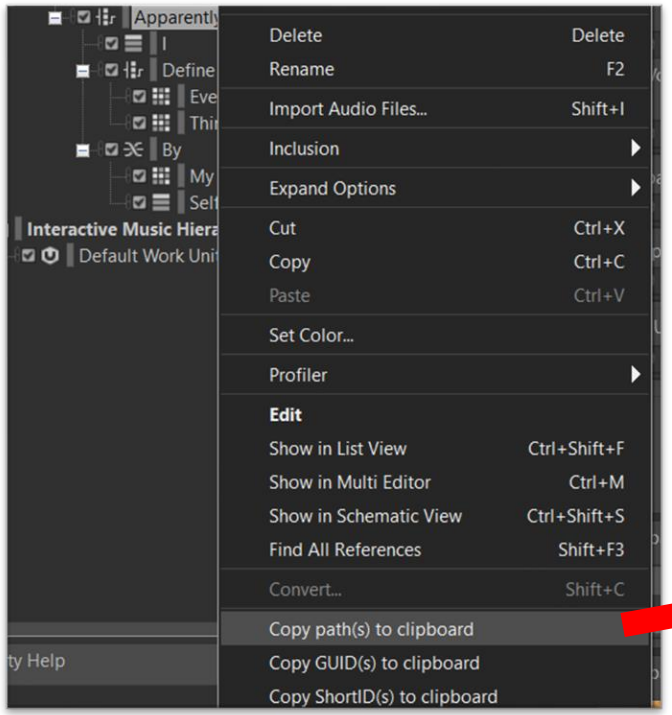
If the contents of the drop-down list are inconsistent with the actual contents in Wwise, you need to click the "Refresh/Reset" button below to refresh the drop-down list before making a selection.



Set It and Forget It! Configure Automation Templates!

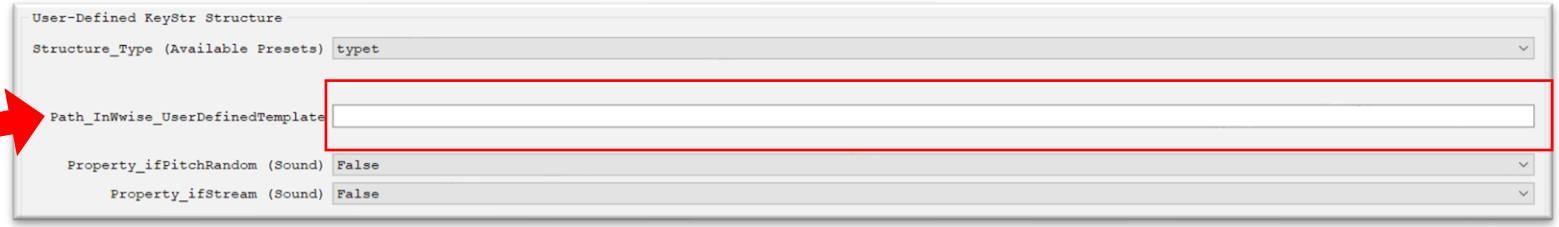
Filling Example: typet has fewer attributes, one of which is different from all other types

Path_InWwise_UserDefinedTemplate:
Here, you fill in the template path created by the user in Wwise. This path needs to be copied and pasted from Wwise.



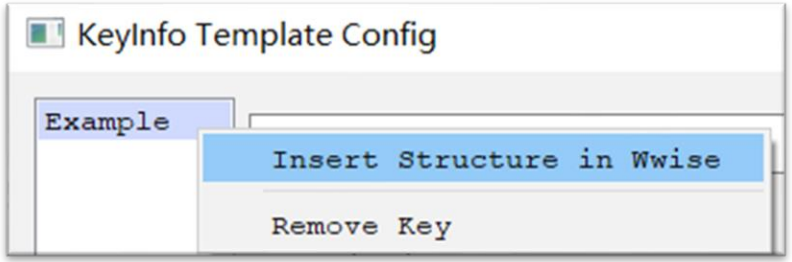
Method to copy the path:

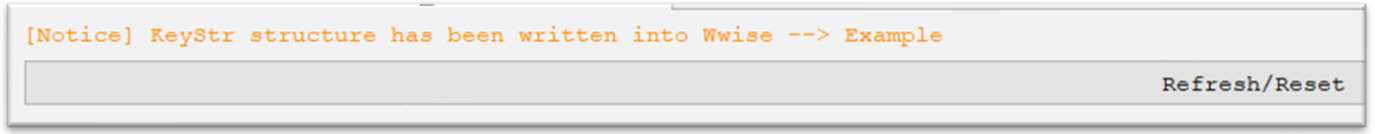
1) Press and hold the Shift key on the keyboard.
2) Right-click the target container object in Wwise.
3) Select Copy path(s) to clipboard.
4) Release the Shift key and paste the path.



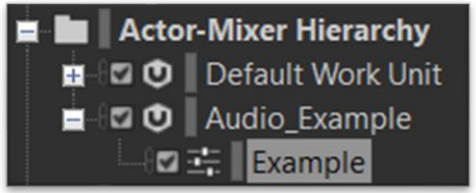
Set It and Forget It! Configure Automation Templates!

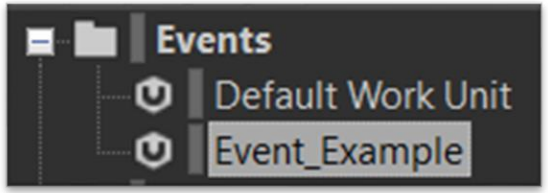
After setting the attributes, right-click the newly created KeyStr in the list on the left and select "Insert Structure in Wwise". Wait for the writing prompt at the bottom of the window.

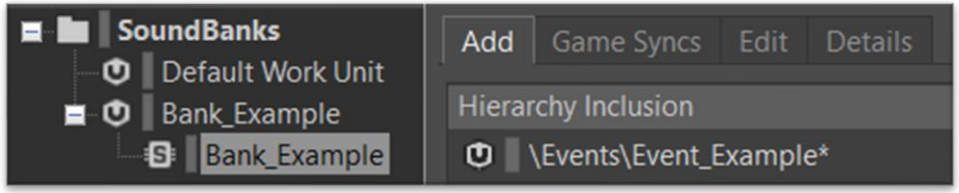




Return to Wwise and confirm the writing results. If there are no issues, the process is complete.







Reminder:

We can see that each keyword corresponds to 3 WorkUnits (Actor-Mixer, Events, SoundBanks), which are in harmony and neatly organized. At the same time, **each KeyStr actually corresponds to a bnk file as well**. Therefore, planning KeyStr according to the granularity of SoundBanks is a **highly recommended** practice!

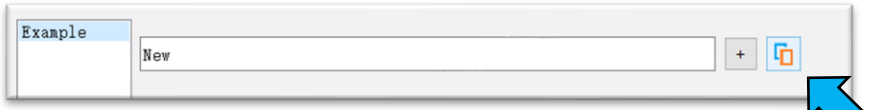
The granularity of KeyStr should be kept in sync with the granularity of bnk files, the finer the better. Avoid arbitrary classification and naming as much as possible.

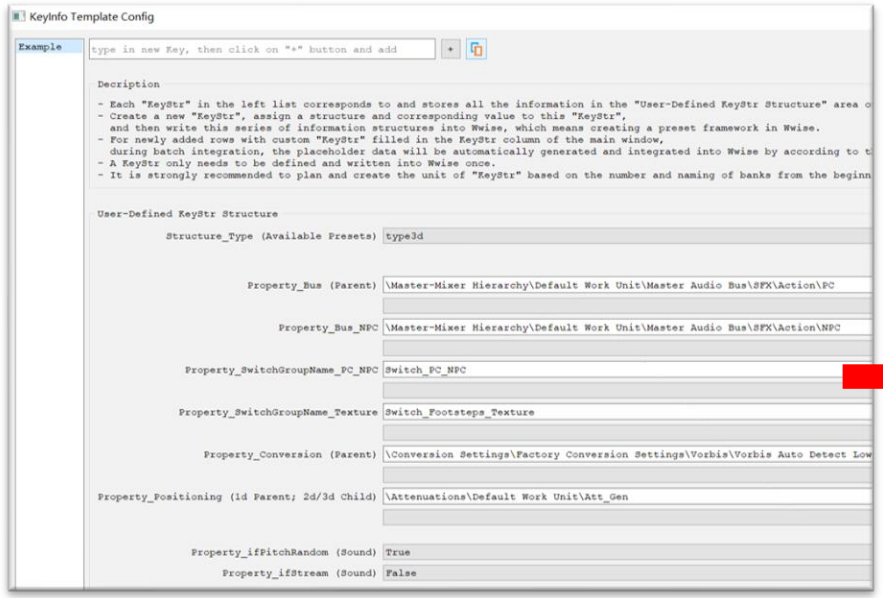
Set It and Forget It! Configure Automation Templates!

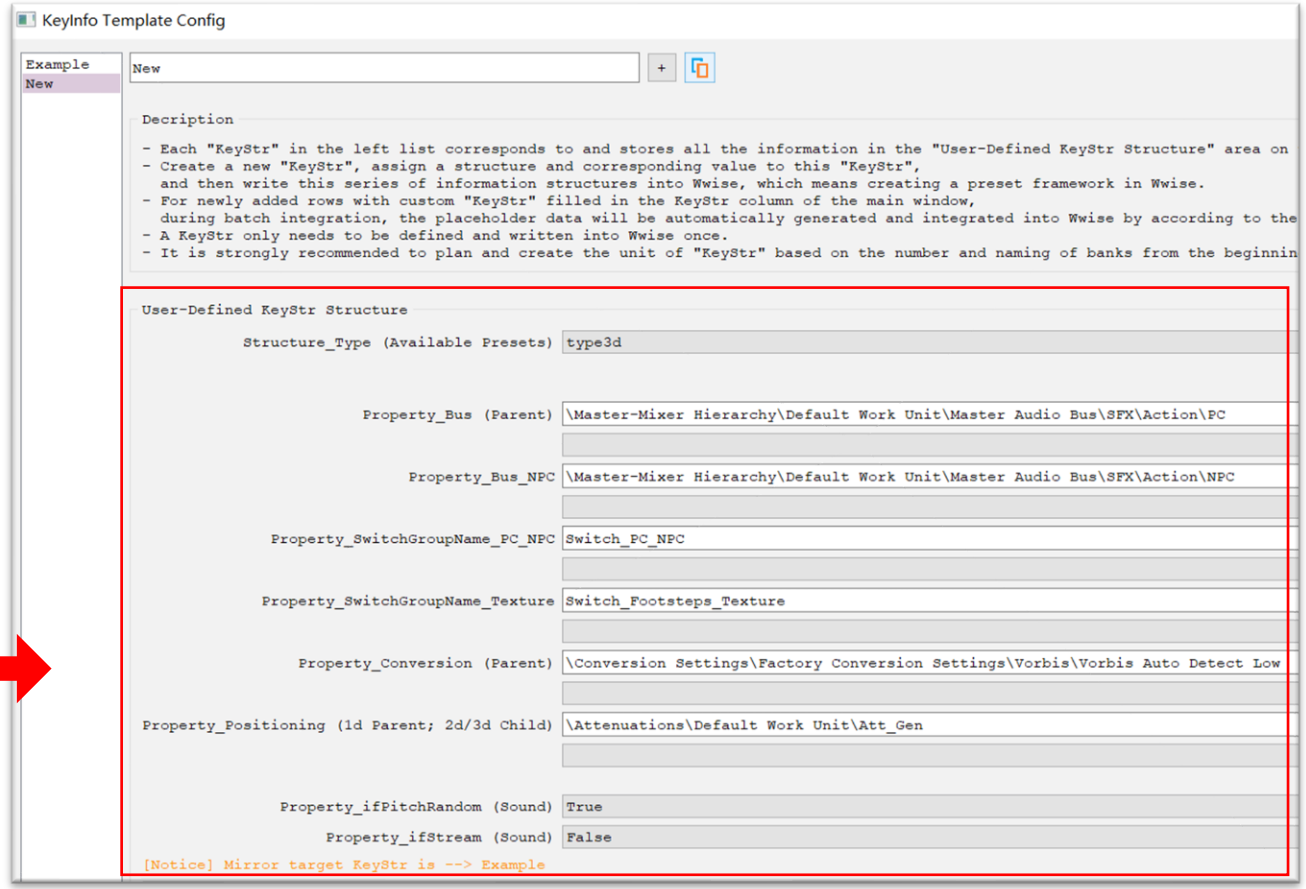
Of course, if you have to go through the entire process every time you create a new keyword, it would be very troublesome. Therefore, to the right of the "+" button, there is also a **MIRROR** button.

For example: Select the KeyStr "Example" to be mirrored in the list on the left, enter the new KeyStr "New" in the newly added text box, and then click the **MIRROR** button. At this point, all the attributes of the newly added KeyStr "New" will automatically display the exact same content as "Example". If there is nothing to adjust, you can directly right-click to write it into Wwise.

With that, the process of configuring the automation template is complete. The **KeyStr** can now **participate in the naming in the main interface.**^_^





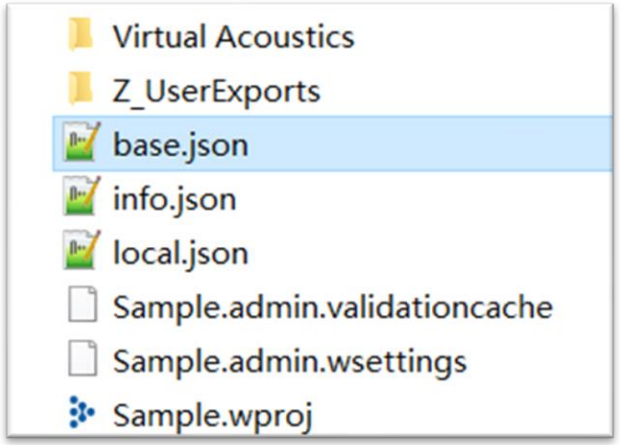


Set It and Forget It! Configure Automation Templates!

【Additional Tips】

All data in the Configure Automation Template window will be recorded in the base.json file in the current Wwise project directory. Text changes in this window will be automatically saved when changes are detected.

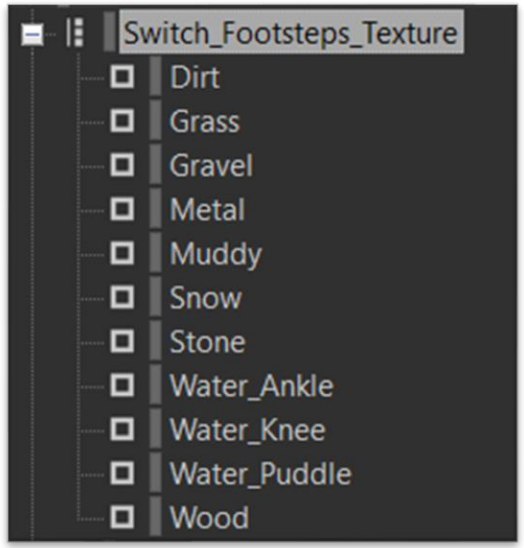
Note: For data security and plugin stability, please modify the content through the plugin window and try not to manually change the JSON file directly!

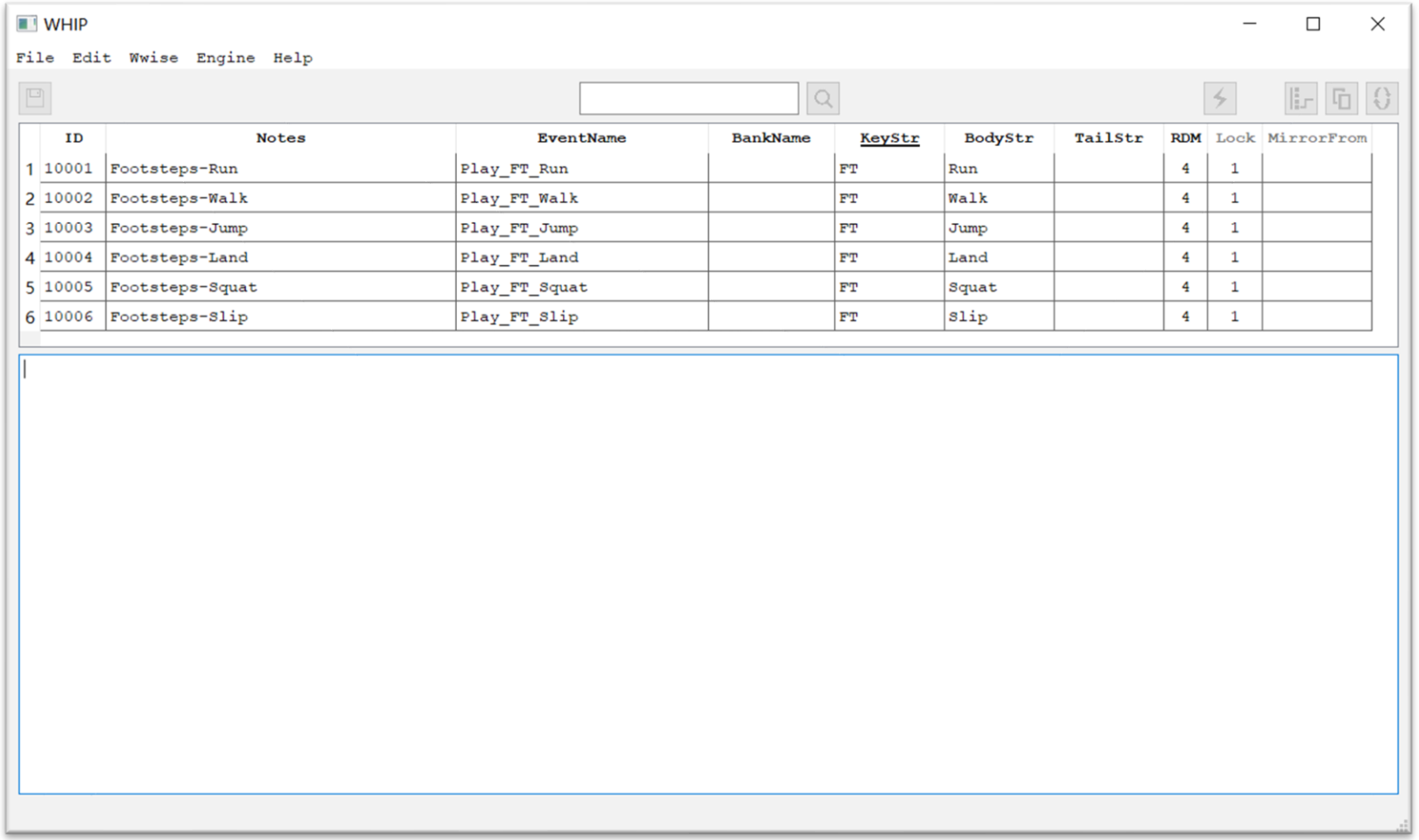



```
{
    "$ProjectStr$": "Sample.wproj",
    "--------------[Key Info]-------------": "----------------------",
    "Data_KeyInfo": {
        "Act": {
            "Structure_Type": "type2d",
            "Path_InWwise_UserDefinedTemplate": "",
            "Path_File_PlaceholderWAV": "cf\\wavPlaceholder\\silence.wav",
            "Path_Folder_TargetWAV": "Originals\\SFX\\",
            "Path_InWwise_TargetActorMixer": "\\Actor-Mixer Hierarchy\\Audio_Act\\Act",
            "Path_InWwise_TargetEvent": "\\Events\\Event_Act",
            "Path_InWwise_TargetBank": "\\SoundBanks\\Bank_Act\\Bank_Act",
            "Property_Conversion": "\\Conversion Settings\\Factory Conversion Settings\\Vorbi
            "Property_Positioning": "\\Attenuations\\Default Work Unit\\Att_Ally",
            "Property_Bus": "\\Master-Mixer Hierarchy\\Default Work Unit\\Master Audio Bus\\S
            "Property_Bus_NPC": "\\Master-Mixer Hierarchy\\Default Work Unit\\Master Audio Bu
            "Property_SwitchGroupName_PC_NPC": "Switch_PC_NPC",
            "Property_SwitchGroupName_Texture": "",
            "Property_ifPitchRandom": "False",
            "Property_ifStream": "False"
        },
        "UI": {
            "Structure_Type": "type1d",
            "Path_InWwise_UserDefinedTemplate": "",
            "Path_File_PlaceholderWAV": "cf\\wavPlaceholder\\silence.wav",
            "Path_Folder_TargetWAV": "Originals\\SFX\\",
            "Path_InWwise_TargetActorMixer": "\\Actor-Mixer Hierarchy\\Audio_UI\\UI",
            "Path_InWwise_TargetEvent": "\\Events\\Event_UI",
            "Path_InWwise_TargetBank": "\\SoundBanks\\Bank_UI\\Bank_UI",
            "Property_Conversion": "",
            "Property_Positioning": "",
```

Ground materials have increased! Many containers involving material switches need to be expanded, what should I do?

Assumption: The "Switch_Footsteps_Texture" object needs to add a "Sand", but it affects 6 previously integrated objects.

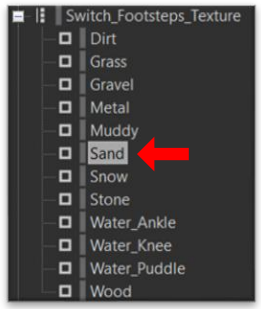


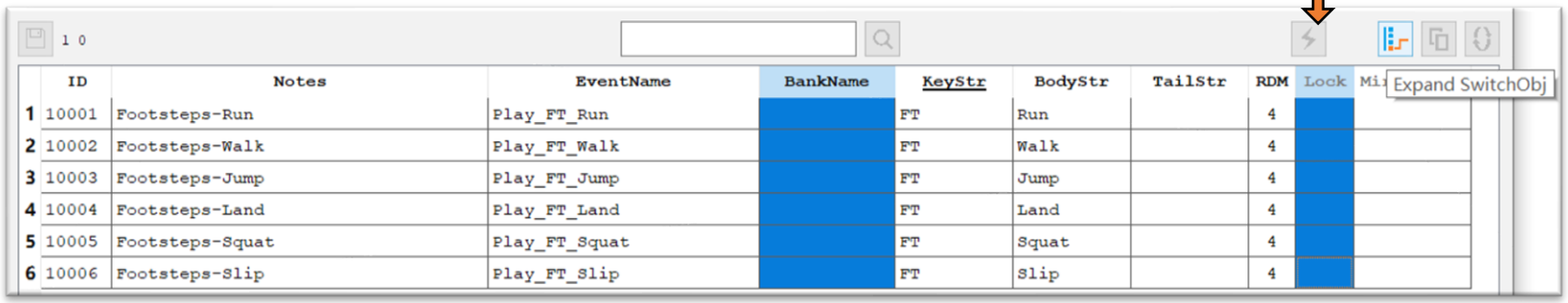


	ID	Notes	EventName	BankName	KeyStr	BodyStr	TailStr	RDM	Lock	MirrorFrom
1	10001	Footsteps-Run	Play_FT_Run		FT	Run		4	1	
2	10002	Footsteps-Walk	Play_FT_Walk		FT	Walk		4	1	
3	10003	Footsteps-Jump	Play_FT_Jump		FT	Jump		4	1	
4	10004	Footsteps-Land	Play_FT_Land		FT	Land		4	1	
5	10005	Footsteps-Squat	Play_FT_Squat		FT	Squat		4	1	
6	10006	Footsteps-Slip	Play_FT_Slip		FT	Slip		4	1	

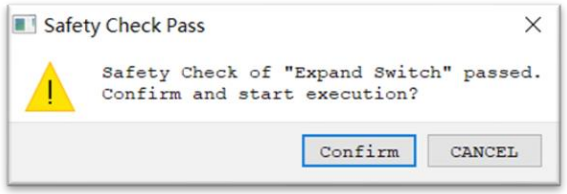
Ground materials have increased! Many containers involving material switches need to be expanded, what should I do?

Step 1: In Wwise, add the "Sand" object to "Switch_Footsteps_Texture" and save.
Step 2: For all rows in the table that involve adding switches, clear the contents of the BankName column to help identify the new content after the second integration, and then clear the contents of the Lock column to unlock. Save the table.
Step 3: Select the row to be processed (any cell will do) and click the **"Expand SwitchObj"** button. If the safety pre-check passes, click confirm to start execution. If it does not pass, fix the problem and click again.





	ID	Notes	EventName	BankName	KeyStr	BodyStr	TailStr	RDM	Lock	Mi
1	10001	Footsteps-Run	Play_FT_Run		FT	Run		4		
2	10002	Footsteps-Walk	Play_FT_Walk		FT	Walk		4		
3	10003	Footsteps-Jump	Play_FT_Jump		FT	Jump		4		
4	10004	Footsteps-Land	Play_FT_Land		FT	Land		4		
5	10005	Footsteps-Squat	Play_FT_Squat		FT	Squat		4		
6	10006	Footsteps-Slip	Play_FT_Slip		FT	Slip		4		



Step 4: After execution, pay attention to the changes in the EventName, BankName, and Lock columns in the table. Open Wwise to confirm the integration results. Save the table data, replace the temporary silence placeholders with the official resources, and the process ends.

	ID	Notes	EventName	BankName	KeyStr	BodyStr	TailStr	RDM	Lock	MirrorFrom
1	10001	Footsteps-Run	Play_FT_Run	Bank_FT	FT	Run		4	1	
2	10002	Footsteps-Walk	Play_FT_Walk	Bank_FT	FT	Walk		4	1	
3	10003	Footsteps-Jump	Play_FT_Jump	Bank_FT	FT	Jump		4	1	
4	10004	Footsteps-Land	Play_FT_Land	Bank_FT	FT	Land		4	1	
5	10005	Footsteps-Squat	Play_FT_Squat	Bank_FT	FT	Squat		4	1	
6	10006	Footsteps-Slip	Play_FT_Slip	Bank_FT	FT	Slip		4	1	

Ground materials have increased! Many containers involving material switches need to be expanded, what should I do?

View Results:

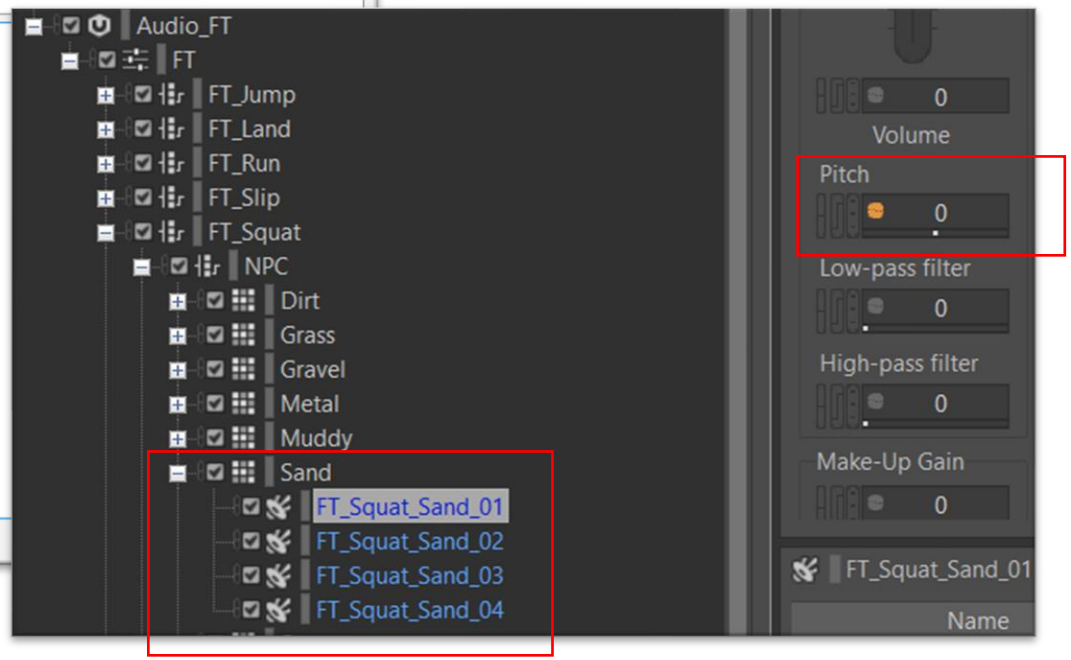
WHIP

File Edit Wwise Engine Help

13 0

	ID	Notes	EventName	BankName	KeyStr	BodyStr	TailStr	RDM	Lock	MirrorFrom
1	10001	Footsteps-Run	Play_FT_Run	Bank_FT	FT	Run		4	1	
2	10002	Footsteps-Walk	Play_FT_Walk	Bank_FT	FT	Walk		4	1	
3	10003	Footsteps-Jump	Play_FT_Jump	Bank_FT	FT	Jump		4	1	
4	10004	Footsteps-Land	Play_FT_Land	Bank_FT	FT	Land		4	1	
5	10005	Footsteps-Squat	Play_FT_Squat	Bank_FT	FT	Squat		4	1	
6	10006	Footsteps-Slip	Play_FT_Slip	Bank_FT	FT	Slip		4	1	

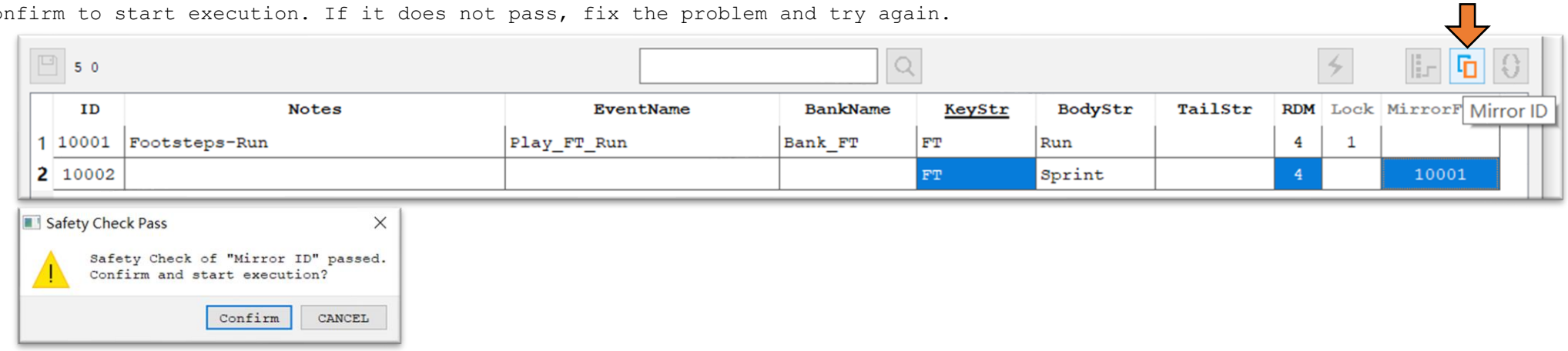
```
Line5: ObjectRef has been found! --> RECORDED
Safety Check PASS! Continue next step...
PC ----------------------------------------> Different Object Detected!
Sand -----> Created!
Sand -----> Assigned!
All New Containers are created and assigned! Ready to create WAV...
NPC ----------------------------------------> Different Object Detected!
Sand -----> Created!
Sand -----> Assigned!
All New Containers are created and assigned! Ready to create WAV...
Line5: New Object has been created and been assigned!
Type Confirmed ----> type3d
Line5: Target Pair Info has been found! --> RECORDED
[Done] C:\Users\admin\Desktop\Sample\Originals\SFX\FT_Squat_Sand_01.wav is created successfully
[Done] C:\Users\admin\Desktop\Sample\Originals\SFX\FT_Squat_Sand_03.wav is created successfully
[Done] C:\Users\admin\Desktop\Sample\Originals\SFX\FT_Squat_Sand_04.wav is created successfully
[Done] C:\Users\admin\Desktop\Sample\Originals\SFX\FT_Squat_Sand_02.wav is created successfully
Line5: WAV Created Successfully --> Done
Line5: is generating WWISE DATA --> ON GOING...
FT_Squat_Sand_01 -----> WAV Imported
FT_Squat_Sand_02 -----> WAV Imported
FT_Squat_Sand_03 -----> WAV Imported
FT_Squat_Sand_04 -----> WAV Imported
```



Want to completely reuse an Event, but the containers and resource naming from inside to outside all need to be brand new! What should I do?

Assumption: We want to reuse the content of "Footsteps-Run" and mirror it into a set of "Footsteps- Sprint" content, but the naming needs to be changed from head to toe.

Step 1: In the table, add a new row with ID 10002, and in the KeyStr column cell, fill in the same KeyStr as the row with ID 10001; the number in the RDM cell may also need to be the same.
Step 2: In the MirrorFrom cell of the new row, fill in the ID of the mirroring target: 10001. Save the table.
Step 3: Select the row to be processed (any cell will do) and click the "**Mirror ID**" button. If the safety pre-check passes, click confirm to start execution. If it does not pass, fix the problem and try again.



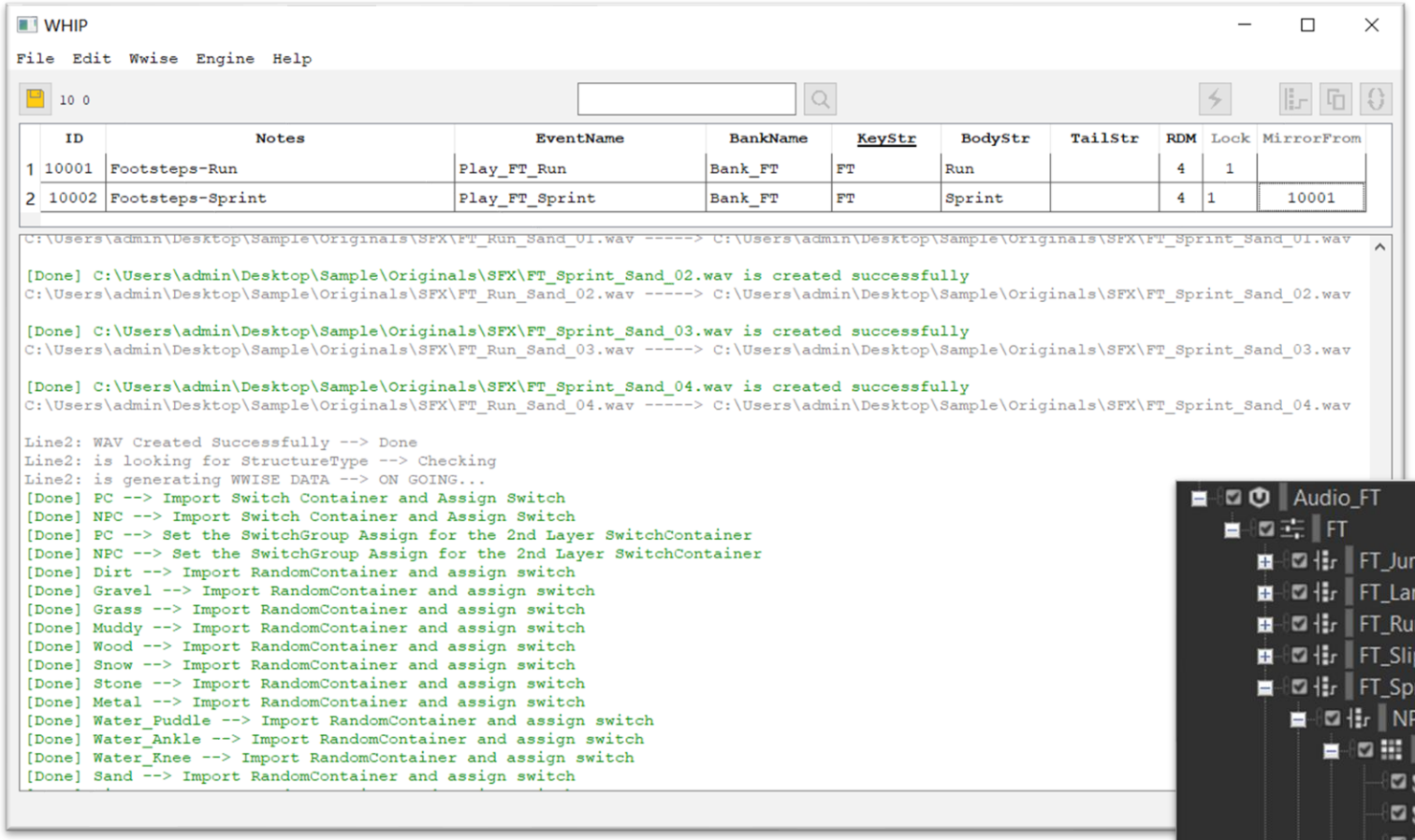
	ID	Notes	EventName	BankName	KeyStr	BodyStr	TailStr	RDM	Lock	MirrorF
1	10001	Footsteps-Run	Play_FT_Run	Bank_FT	FT	Run		4	1	
2	10002				FT	Sprint		4		10001

Step 4: After execution, pay attention to the changes in the EventName, BankName, and Lock columns in the table. Open Wwise to confirm the integration results. Save the table data, replace the temporary silence placeholders with the official resources, and the process ends.

	ID	Notes	EventName	BankName	KeyStr	BodyStr	TailStr	RDM	Lock	MirrorFrom
1	10001	Footsteps-Run	Play_FT_Run	Bank_FT	FT	Run		4	1	
2	10002		Play_FT_Sprint	Bank_FT	FT	Sprint		4	1	10001

Want to completely reuse an Event, but the containers and resource naming from inside to outside all need to be brand new! What should I do?

View Results:

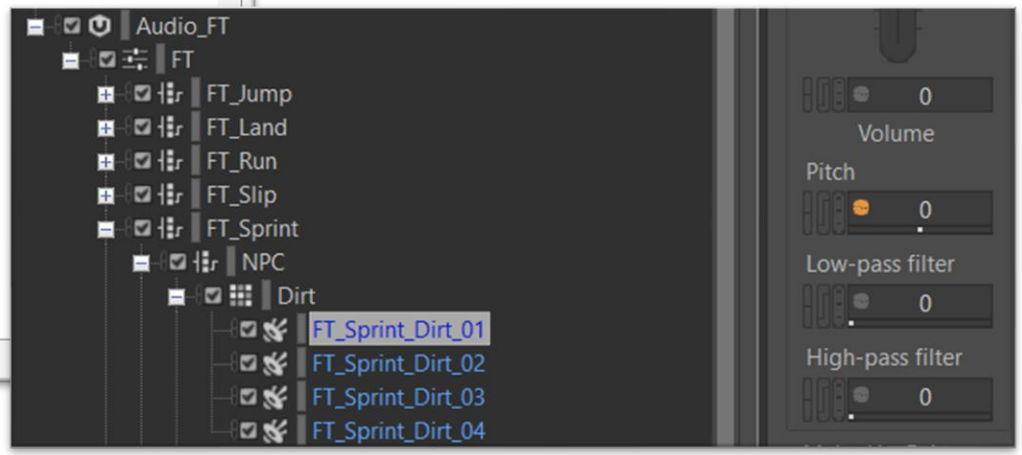


Reminder:

The content of 10002 is completely based on mirroring 10001 and then modifying the naming.

The KeyStr in the naming of the new object **must be consistent with the mirroring target.**

For type1d and type2d, the random number can be different from the mirroring target, **but for type3d, the random number must be the same.**

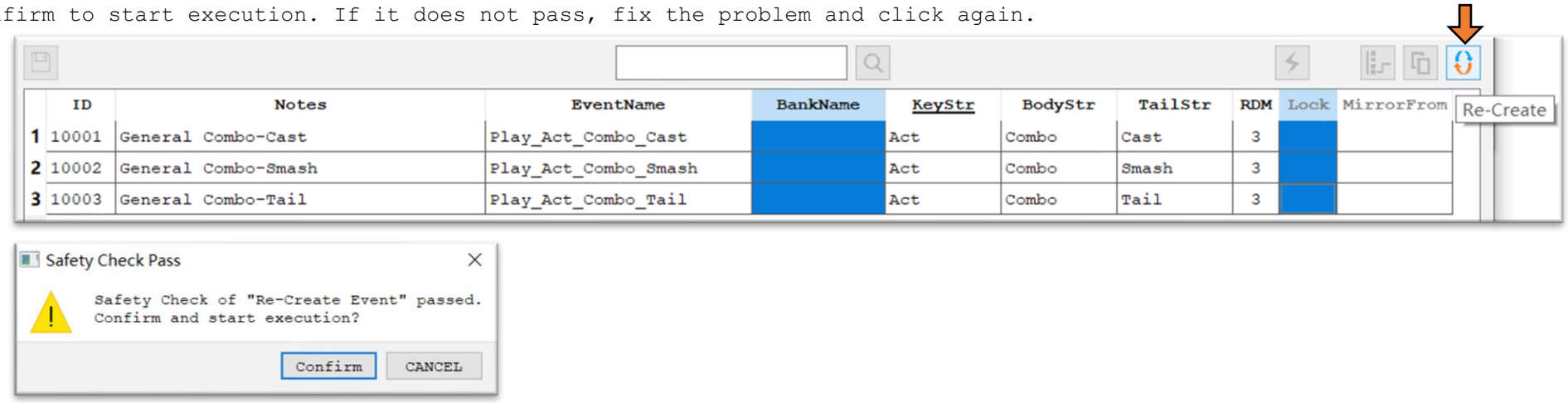


Start from scratch! The effects generated by AI in the previous round were mediocre, do it again! (Under development, only a reference)

Assumption: We just generated a batch of skill materials through AI one-click integration, but after testing, we feel it's not quite right. We want to regenerate a new set through AI **without adding new objects or changing the naming**.

Step 1: For all rows in the table that involve adding switches, clear the contents of the **BankName column** to help identify the new content after the second integration, and then clear the contents of the **Lock column** to unlock. Save the table.

Step 2: Select the row to be processed (any cell will do) and click the "**Re-Create**" button. If the safety pre-check passes, click confirm to start execution. If it does not pass, fix the problem and click again.



	ID	Notes	EventName	BankName	KeyStr	BodyStr	TailStr	RDM	Lock	MirrorFrom
1	10001	General Combo-Cast	Play_Act_Combo_Cast		Act	Combo	Cast	3		
2	10002	General Combo-Smash	Play_Act_Combo_Smash		Act	Combo	Smash	3		
3	10003	General Combo-Tail	Play_Act_Combo_Tail		Act	Combo	Tail	3		

Step 3: After execution, pay attention to the changes in the EventName, BankName, and Lock columns in the table. Open Wwise to confirm the integration results. Save the table data, and the process ends.

	ID	Notes	EventName	BankName	KeyStr	BodyStr	TailStr	RDM	Lock	MirrorFrom
1	10001	General Combo-Cast	Play_Act_Combo_Cast	Bank_Act	Act	Combo	Cast	3	1	
2	10002	General Combo-Smash	Play_Act_Combo_Smash	Bank_Act	Act	Combo	Smash	3	1	
3	10003	General Combo-Tail	Play_Act_Combo_Tail	Bank_Act	Act	Combo	Tail	3	1	

Start from scratch! The effects generated by AI in the previous round were mediocre, do it again! (Under development, only a reference)

View Results:

WHIP

File Edit Wwise Engine Help

	ID	Notes	EventName	BankName	KeyStr	BodyStr	TailStr	RDM	Lock	MirrorFrom
1	10001	General Combo-Cast	Play_Act_Combo_Cast	Bank_Act	Act	Combo	Cast	3	1	
2	10002	General Combo-Smash	Play_Act_Combo_Smash	Bank_Act	Act	Combo	Smash	3	1	
3	10003	General Combo-Tail	Play_Act_Combo_Tail	Bank_Act	Act	Combo	Tail	3	1	

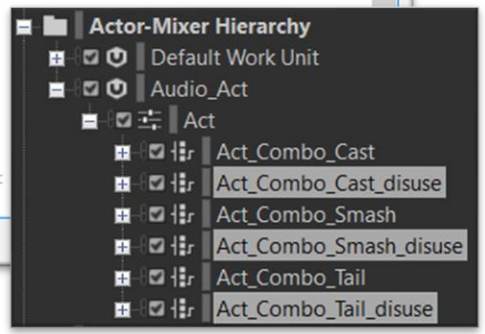
```
[Done] Act_Combo_Smash_01 --> Import WAV for RandomContainer
[Done] Act_Combo_Smash_02 --> Import WAV for RandomContainer
[Done] Act_Combo_Smash_03 --> Import WAV for RandomContainer
[Done] Act_Combo_Smash_01 --> Import WAV for RandomContainer
[Done] Act_Combo_Smash_02 --> Import WAV for RandomContainer
[Done] Act_Combo_Smash_03 --> Import WAV for RandomContainer
Line2: WWISE implement --> DONE.
Line2: EventName and LockMark have been filled in TABLE --> DONE.
Line3: is a new (or an unlocked) object
Line3: ObjectRef has been found! --> RECORDED
Line3: is creating WAVs --> Checking
Naming Check PASSED ------> Ready to create WAV!
Line3: WAV Created Successfully --> Done
Line3: is looking for StructureType --> Checking
Line3: is generating WWISE DATA --> ON GOING...
[Done] PC --> Import RandomContainer and assign switch
[Done] NPC --> Import RandomContainer and assign switch
[Done] Act_Combo_Tail_01 --> Import WAV for RandomContainer
[Done] Act_Combo_Tail_02 --> Import WAV for RandomContainer
[Done] Act_Combo_Tail_03 --> Import WAV for RandomContainer
[Done] Act_Combo_Tail_01 --> Import WAV for RandomContainer
[Done] Act_Combo_Tail_02 --> Import WAV for RandomContainer
[Done] Act_Combo_Tail_03 --> Import WAV for RandomContainer
Line3: WWISE implement --> DONE.
Line3: EventName and LockMark have been filled in TABLE --> DONE.

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> ALL FINISHED! <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

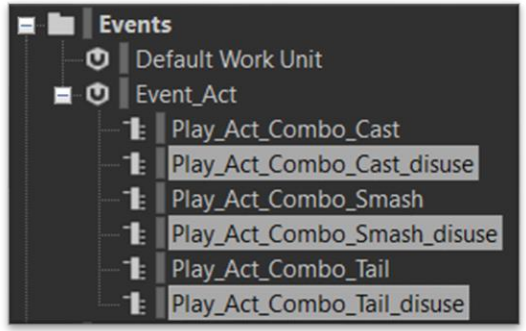
Reminder:

Since the current 2.0 version does not include the function of AI-generated placeholders, this function can only achieve the effect of "simply reintegrating once" for the time being.

The previous old content will not be deleted, but "_disuse" will be added after the original name and kept in the project for reference and comparison with the new content.

Also: If this function is used to deal with the scenario of material increase, the effect is the same as "Expand SwitchObj".

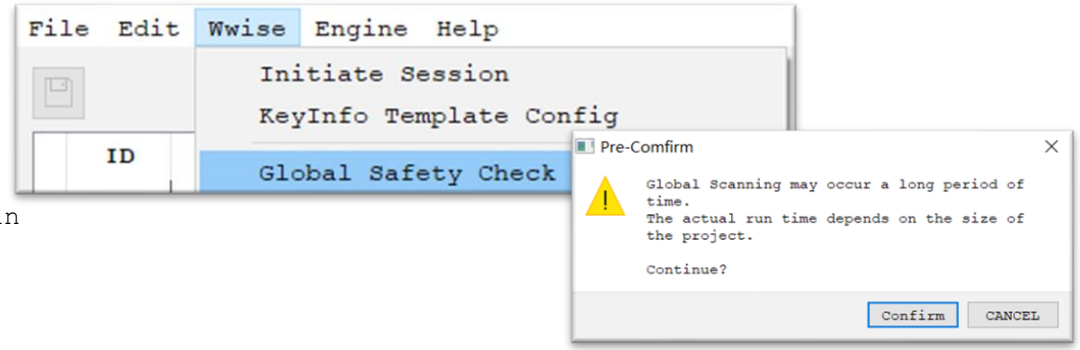




Global Safety Check!

Are there any container objects that are "not yet assigned to an Event"?
Are there any Event events that are "not written to the table"?
Are there any automation paths that "do not belong to the current Wwise project"?
Are there any objects that "do not comply with the structure specifications"?
...

Operation steps:
Click "Wwise→Global Safety Check" in the menu bar, and you can obtain a detailed inspection report in the log console information! For example, see the image below:



	ID	Notes	EventName	BankName	KeyStr	BodyStr	TailStr	RDM	Lock	MirrorFrom
1	10001	General Combo-Cast	Play_Act_Combo_Cast	Bank_Act	Act	Combo	Cast	3	1	
2	10002	General Combo-Smash	Play_Act_Combo_Smash	Bank_Act	Act	Combo	Smash	3	1	
3	10003	General Combo-Tail	Play_Act_Combo_Tail	Bank_Act	Act	Combo	Tail	3	1	

```
>>>>>>>>>>>>>>>>>>>> Global Safety Check - Start <<<<<<<<<<<<<<<<<<<<<<<

[Unused Event] --> {'Play_FT_Gen': '{CD7EA7C5-322A-4FD7-B05D-867BE54BD072}'}
[Unused Event] --> {'Play_UI_Equip_LevelUp': '{4FB0645F-383D-49A6-A195-899A427CA953}'}
[Unused Event] --> {'Play_Special_Amazing': '{1A57C482-F822-4AFA-83FB-E23DF596FCF7}'}
[Unused Event] --> {'Play_FT_Sprint': '{881B5D05-DF57-47AF-88DD-EB1A0FFF16CC}'}
[Unused Event] --> {'Play_FT_Jump': '{ADB4A688-7AB3-46E1-BBA1-F2AD9D661D97}'}
[Unused Event] --> {'Play_Act_Combo_Tail_disuse': '{406D8D7D-96C3-4FDC-9B5B-15497DD01F6E}'}
[Unused Event] --> {'Play_FT_Run': '{7B767C95-D70D-40DD-BD3A-05517157BB1B}'}
[Unused Event] --> {'Play_FT_Squat': '{2EA69C2D-EF33-437C-A8AF-45DCEA2DDCB3}'}
[Unused Event] --> {'Play_Act_Combo_Cast_disuse': '{9903928F-4B07-442D-AC5C-51EBFB465D44}'}
[Unused Event] --> {'Play_Act_Combo_Smash_disuse': '{B726E9FC-3F16-4EA3-9FFB-841CA97DA1A2}'}
[Unused Event] --> {'Play_FT_Walk': '{AF9E5439-F1C0-443A-866A-DE3D836E2D26}'}
[Unused Event] --> {'Play_FT_Slip': '{467C761D-B877-4562-B5EE-ACDA17EF7D34}'}
[Unused Event] --> {'Play_FT_Land': '{BACCF43D-9BF6-4107-8DBF-869AB2A673FC}'}

>>>>>>>>>>>>>>>>>>>> Global Safety Check - End <<<<<<<<<<<<<<<<<<<<<<<
```

Global Safety Check!

【Supplementary Explanation】

***When generating WWISE data, the plugin follows a series of strict hierarchy specifications:**

-The Audio WorkUnit of each KeyStr will definitely be generated under the \Actor-Mixer Hierarchy root path. (Prohibited from moving, prohibited from changing hierarchy)

-The parent Actor-Mixer of each KeyStr will definitely be generated under the corresponding WorkUnit (allowed to move at the same level, prohibited from changing hierarchy)

-Each ObjectRef container assigned to an Event will definitely be generated under the corresponding \Actor-Mixer Hierarchy\WorkUnit(Audio_XXX)\Actor-Mixer(Audio_XXX) path (allowed to move at the same level, prohibited from changing hierarchy)

-The Event WorkUnit of each KeyStr will definitely be generated under the \Events root path (prohibited from moving, prohibited from changing hierarchy)

-Each Event will be generated under the path of the corresponding Event WorkUnit (allowed to move at the same level, prohibited from changing hierarchy)

-The object generation path specifications for SoundBanks are the same as those for Audio and Event (allowed to move freely, allowed to change hierarchy freely)

***Safety Warning:**

Do not easily change the hierarchy structure and relationship of objects automatically generated by the plugin, and it is strongly recommended to use "Virtual Folder" with caution.

If the hierarchy structure of objects is disrupted, the safety check function will lose its reference system for judgment, resulting in incorrect and meaningless inspection results, and it will also destroy the safety and accuracy of related extended function modules!

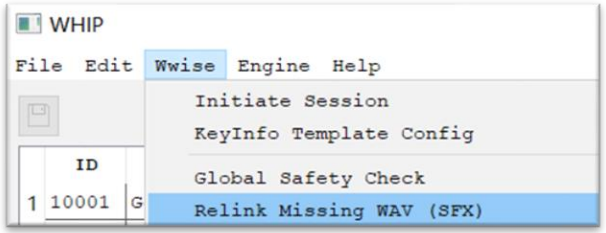
Why are such strict specifications for the hierarchy structure imposed when designing the plugin?

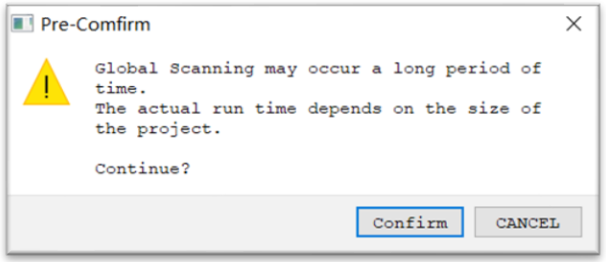
First of all, such restrictive specifications do not hinder the "creative development of audio content" at all. It has two main purposes:
1) To ensure data security, readability, and maintainability during multi-person collaboration.
2) To lay the foundation for extended function modules, considering the demand for further attempts to expand the boundaries of automation capabilities. For example, the conditional basis for "automatic debug" in the future.

Regarding multi-person collaboration. Precisely because Wwise allows objects at any level to be assigned to an Event, we need to be more vigilant about the unnecessary time waste and trouble that "disorder" may cause! For example, we can use a Switch container to create a Play Event, but any container subordinate to this Switch container can also directly create a Play Event! This freedom seems very large, right? But there are two sides to everything.

WAV is MISSING! What should I do?

Sometimes, we want to move the wav files under the Originals/SFX path, but once we move them, it will cause disconnection in Wwise. If you want to reconnect, you can click "Wwise→Relink Missing WAV(SFX)" in the menu bar. After confirming the pop-up prompt, Wwise will perform a full scan and reconnect, and then reload Wwise.






```
[Alert] Found Missing WAV --> NewTarget\Act_Combo_Cast_01.wav
[Done] Relocate Missing WAV --> Act_Combo_Cast_01.wav
[Alert] Found Missing WAV --> NewTarget\Act_Combo_Cast_02.wav
[Done] Relocate Missing WAV --> Act_Combo_Cast_02.wav
[Alert] Found Missing WAV --> NewTarget\Act_Combo_Cast_03.wav
[Done] Relocate Missing WAV --> Act_Combo_Cast_03.wav
```

Of course, if the disconnection is caused by a change in the wav name, it cannot be automatically reconnected. You may see a prompt similar to the image below.

```
[Alert] Found Missing WAV --> Act_Combo_Cast_01.wav
[FAIL] Can not find possible target
[Alert] Found Missing WAV --> Act_Combo_Cast_02.wav
[FAIL] Can not find possible target
[Alert] Found Missing WAV --> Act_Combo_Cast_03.wav
[FAIL] Can not find possible target
```

Reminder:
In version 2.0 of this plugin, all wav files are generated under the Originals/SFX path, without setting any subfolders, completely flat.

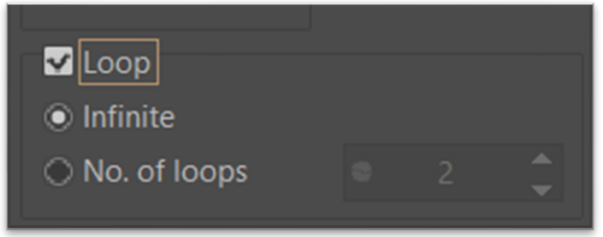
The reason is that under the premise of wav files following a good naming convention, considering factors such as multi-person collaboration, the convenience of overwriting and replacing wav files, the plugin usage process, and the execution complexity of the safety check module, this is the relatively most balanced choice.

Sound objects that need to loop, never miss any!

The only step:

When the end of the naming is the three characters "_LP", after one-click integration, in Wwise, the related Sound objects will automatically check Loop; The plugin data table will automatically generate two Events: one starting with "Play_" and the other starting with "Stop_".

Note: The program only cares about whether the end of the naming is "_LP". Both writing methods of ID 10001 and ID 10002 in the image below are acceptable.



	ID	Notes	EventName	BankName	KeyStr	BodyStr	TailStr	RDM	Lock	MirrorFrom
1	10001				Act	Whoosh_LP		3		
2	10002				Act	Magic	LP	3		

After the integration is completed, two Stop event rows will be automatically added, without the need to manually create them.

	ID	Notes	EventName	BankName	KeyStr	BodyStr	TailStr	RDM	Lock	MirrorFrom
1	10001		Play_Act_Whoosh_LP	Bank_Act	Act	Whoosh_LP		3	1	
2	10002		Play_Act_Magic_LP	Bank_Act	Act	Magic	LP	3	1	
3	10003		Stop_Act_Whoosh_LP						1	
4	10004		Stop_Act_Magic_LP						1	

Sound objects that need to loop, never miss any!

【Discussion】

Why not automatically generate a Stop Event for all Play Events?
Answer: It's not necessary, nor is it needed.

To stop the playback of a certain Event, in addition to using the Stop Event method, the program side can actually directly stop the playback of the corresponding Event by obtaining the playingID of the current Event object and using the stopPlayingID interface, and the way and effect of stopping can also be controlled.

The reason why Stop Events are automatically arranged only for Loop-type sound effects is that usually, the timing and manner in which Loop-type sound effects need to be interrupted are directly arranged by the designer, which is more convenient, of course, still based on the premise of Game Objects. But if we speak frankly, it's actually not necessary.

***Problem discussion:**
In fact, for Events that cannot be attached to a specific Game Object in the game, such as those played directly at map coordinates, it is difficult to stop them through Stop Events. If we detach from Game Objects and directly trigger the same Play Event many times at a certain coordinate (each time with a different playingID), Stop Events may not be able to easily distinguish which specific Play Event object to stop. (Unless Stop Global is used to cut off all Play Events of the same type in the entire scene).

***Safety Warning (off-topic):**
For large-scale projects with a huge amount of data and multi-person collaboration, having multiple different logical instructions in one Event can also be very dangerous.
For example, an Event that includes both Play of a certain object and Stop of a certain object, or even logic such as Set Switch, intertwined together...

Although Wwise kindly provides designers with this "freedom" and "initiative", for projects with multi-person collaboration and maintenance, it is a double side coin.

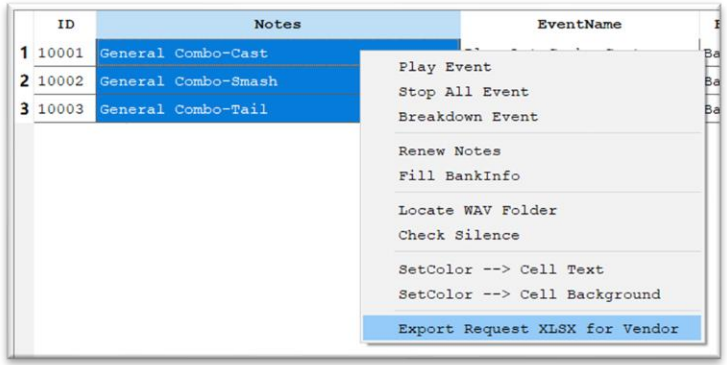
For some logic that is more suitable to be managed by the upstream engine side or client side at the source, based on safety and stability considerations, it is recommended to still hand it over to the upstream for proper management. The content of Events on the Wwise side should be kept as "simple" as possible.

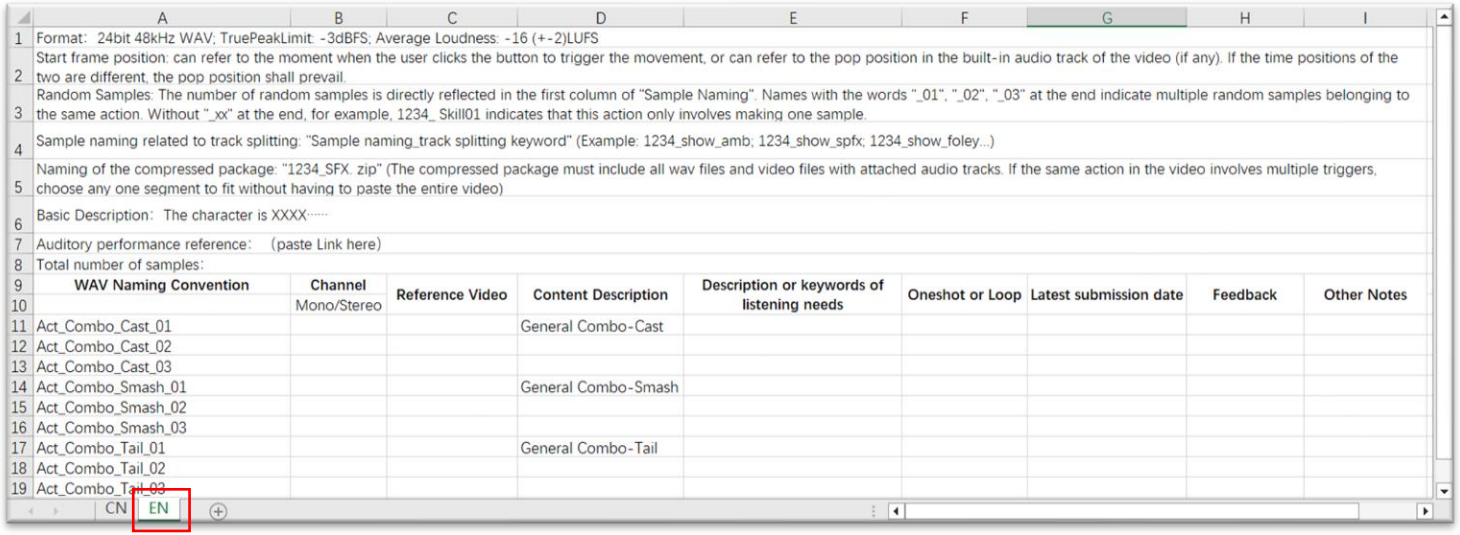
How to export the xlsx requirements table with one click?

The only step:

In the table data on the main interface of the plugin, select the rows that need to be exported. Right-click on the selected cell area, and in the pop-up menu options, click "Export Request XLSX for Vendor". The exported xlsx requirements table is located in the folder path of the project. Double-click to open the requirements table, choose the Chinese or English content as needed, and further edit the content details.

*The name information automatically listed in the "WAV Naming Convention" column of the requirements table corresponds to the name of each WAV. The reason for listing all of them with each random sample as the smallest unit is, on the one hand, to facilitate budget statistics, and on the other hand, to facilitate further cell classification and editing.



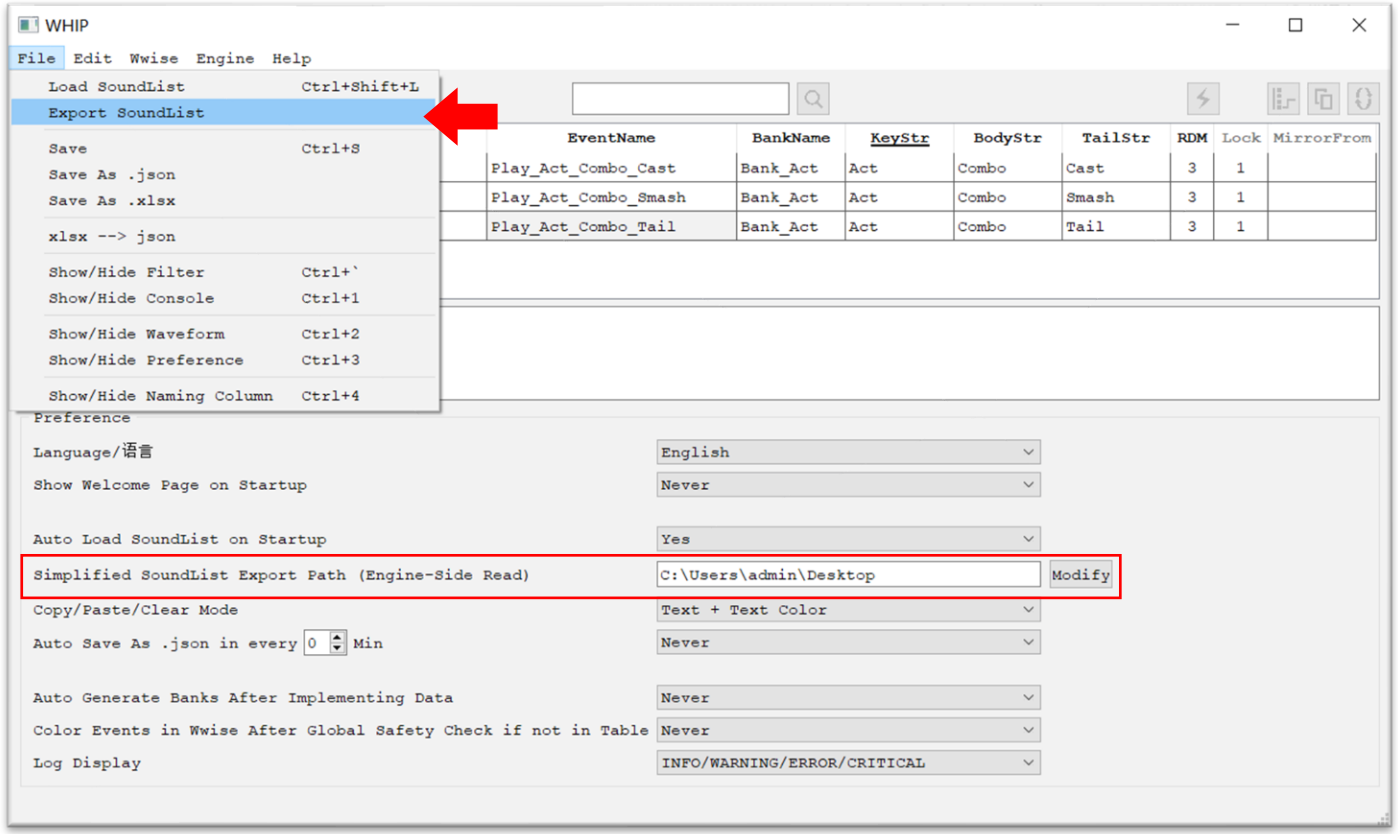


Other Functions&Tips – Export Simplified SoundList

(Click "File - Show/Hide Preferences" in the window top menu bar)

【Export the "Simplified SoundList" for Engine-side reading】

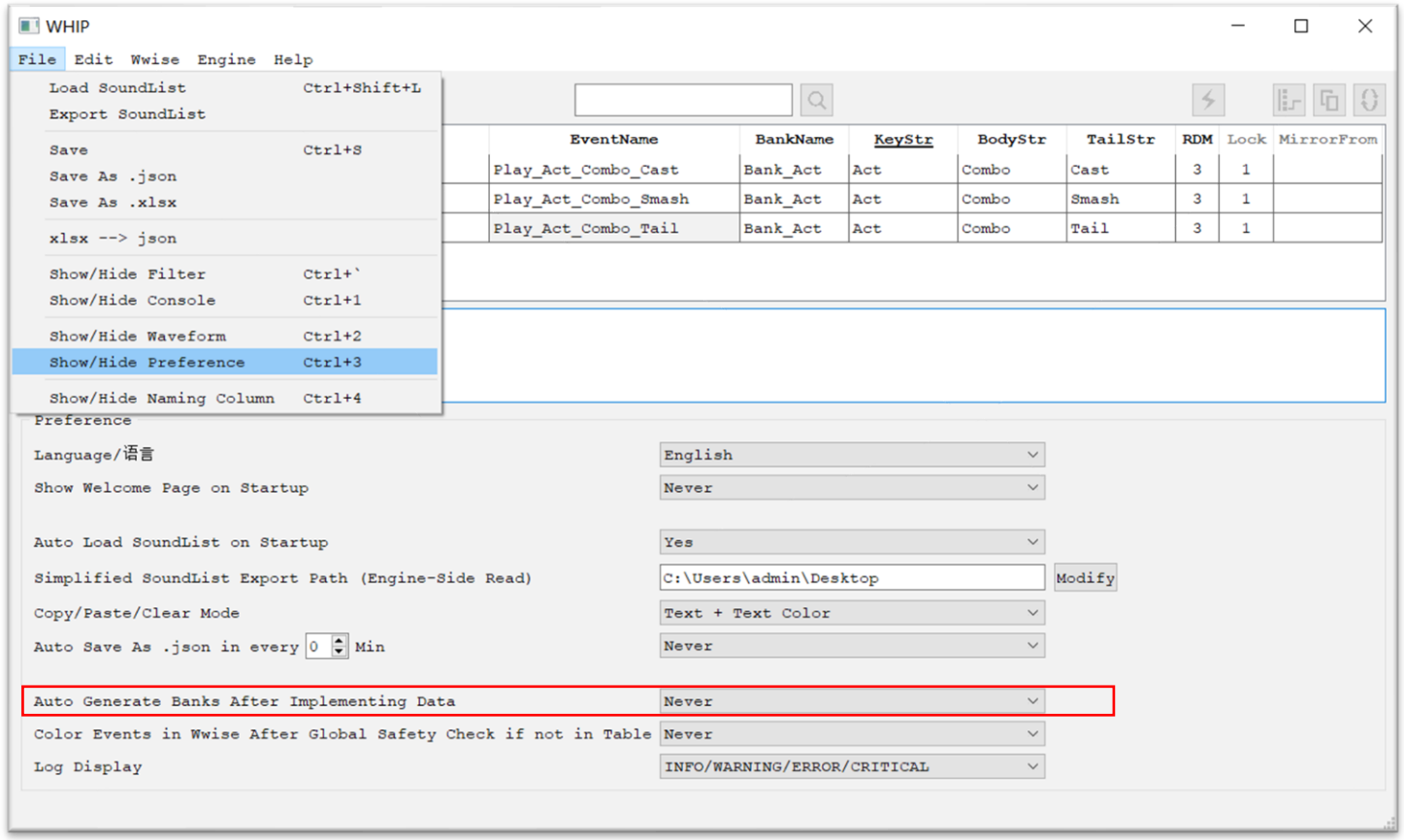
All the information on the main interface is saved in a JSON file, but for Engine-side reading, only three pieces of information are needed: ID, EventName, and BankName. Therefore, we can first specify an engine path in the "Simplified SoundList Export Path (Engine-Side Read)" in the preferences, and then export it through the menu bar "File→Export SoundList".



Other Functions&Tips – Automatically Generate SoundBanks

(Click "File - Show/Hide Preferences" in the window top menu bar)

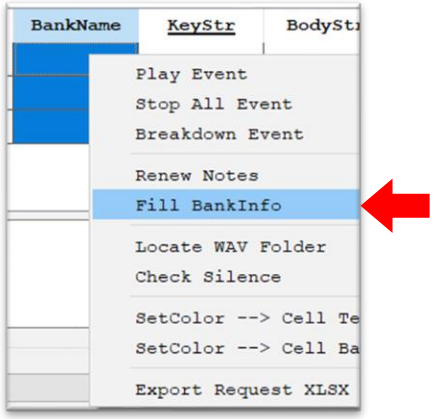
If "Yes" is selected for "Auto Generate Banks After implementing Data", the four function modules "Auto Import Data", "Expand SwitchObj", "Mirror ID", and "Re-Create" will automatically generate SoundBanks after completion and automatically fill the results into the BankName cells in the table. If "Never" is selected, it will not be done automatically.



Reminder:

If the SoundBank is generated manually later, you can select all the rows without BankName information and right-click **"Fill BankInfo"**.

The program will automatically scan **the location** of each Event, and if the SoundBank result is located, it will be automatically filled into the cell.

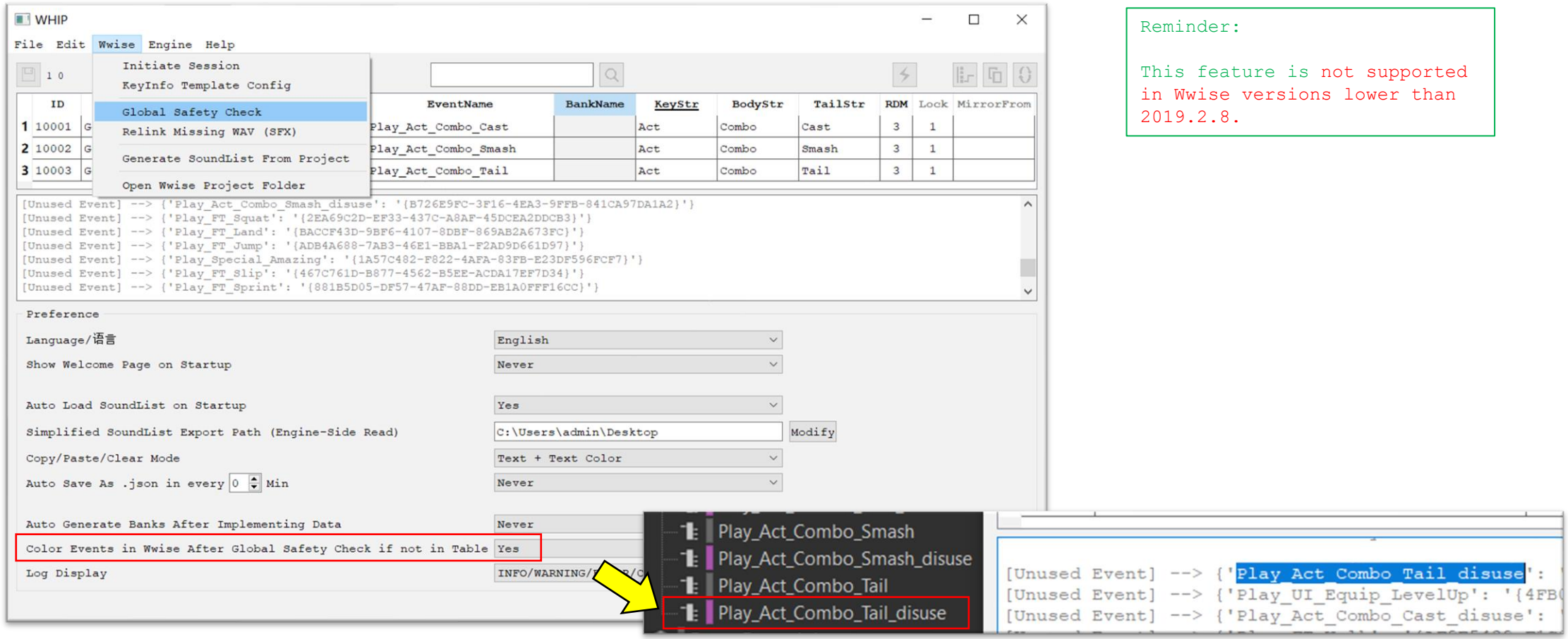


Other Functions&Tips – Set Color For Event

(Click "File - Show/Hide Preferences" in the window top menu bar)

【After full safety check, color the Events in Wwise but not in the cells】

If "Yes" is selected, after clicking "Global Safety Check" in the menu bar, if Events are found in the Wwise project that are not filled in the table, the color of the Events will be automatically marked as **PINK**.



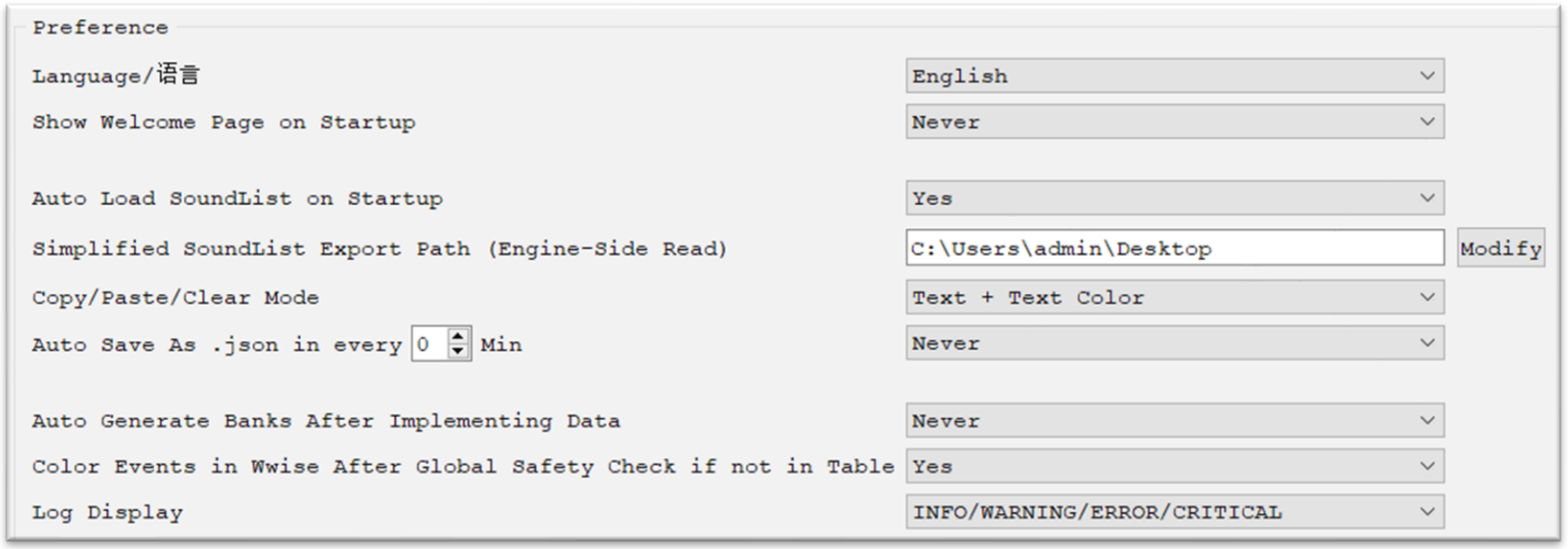
Reminder:

This feature is not supported in Wwise versions lower than 2019.2.8.

Other Functions&Tips – General Preferences

(Click "File - Show/Hide Preferences" in the window top menu bar)

Other functions:



Language/语言: Used to switch the display language of the main interface. Currently, only two options are provided: 中文 (Chinese) and English.

Show Welcome Page on Startup: If "Yes" is selected, the welcome window will pop up every time the plugin launched. If "Never" is selected, it will directly enter the main interface every time the plugin launched.

Auto Load SoundList on Startup: If "Yes" is selected, the audio table JSON file will be automatically loaded at startup and the table will be displayed in the main window; if "Never" is selected, the main window will not load the JSON and display the table after startup, and you need to manually load it through "File→Load SoundList" in the main interface.

Copy/Paste/Clear Mode: According to the content selected by the user, process the text and color information in the cells.

Auto Save As .json in every few minutes: If "Yes" is selected here and a time is set, the tool will automatically save the complete JSON as a snapshot backup at regular intervals based on the set time.

Log display: There are several different levels. The fewer levels selected to display, the fewer types of execution information that can be seen in the log console. It is recommended to keep the level "INFO/WARNING/ERROR/CRITICAL".

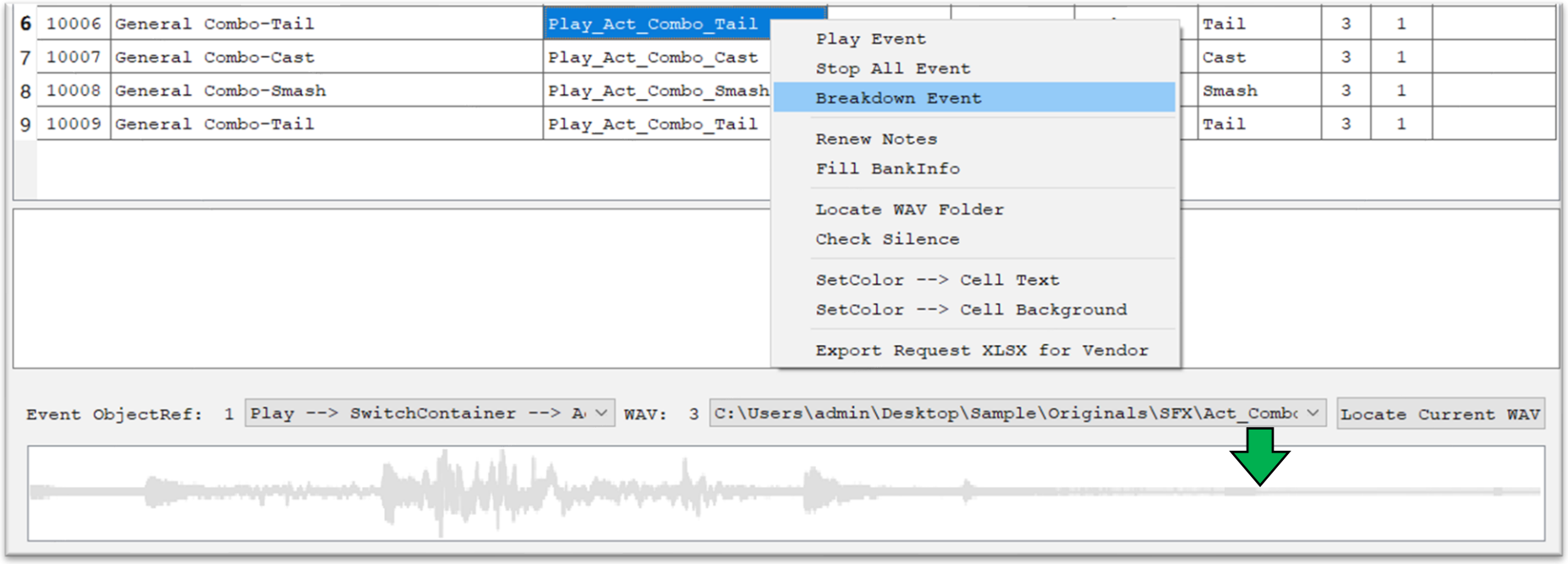
Other Functions&Tips – Quickly Verify Event Status

【Use "Play Event, Stop Event, Breakdown Event" to quickly verify the Event status】

Select EventName in any row, right-click "**Play Event**" or "**Stop Event**", and you can directly play and stop the content of the Event without going back to Wwise to search for it. At this time, Wwise will also directly display the currently playing Event in the foreground.

If you click "**Breakdown Event**", a new area will appear at the bottom of the plugin's main interface. The area will display the Action of the Event and the waveform of the wav corresponding to the ObjectRef in each Action. We can **double-click the left mouse button at any position on the waveform, and it will directly start playing the original wav content from the double-clicked position**. If the wav is long and needs to be stopped in the middle, right-click anywhere on the waveform.

Note: There is a feature not presented in plugin version 2.0: After Breakdown the Event, in addition to displaying the waveform information, it can also display the "playback analysis" of the current Event. For example, it presents a brief analysis report of the factors that affect Event playback in the order of the signal flow from front to back, and prompts the joints that may have bugs.

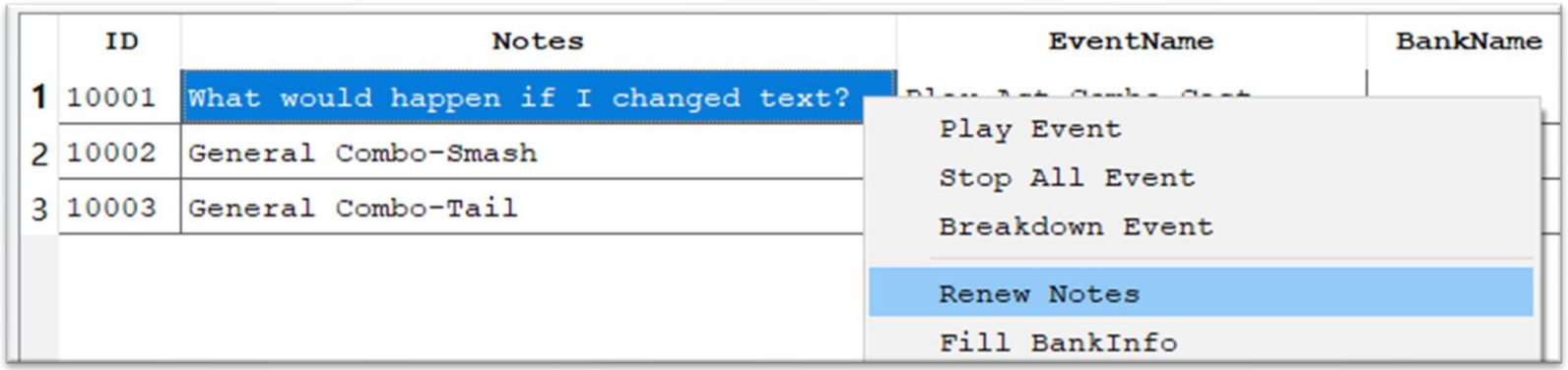


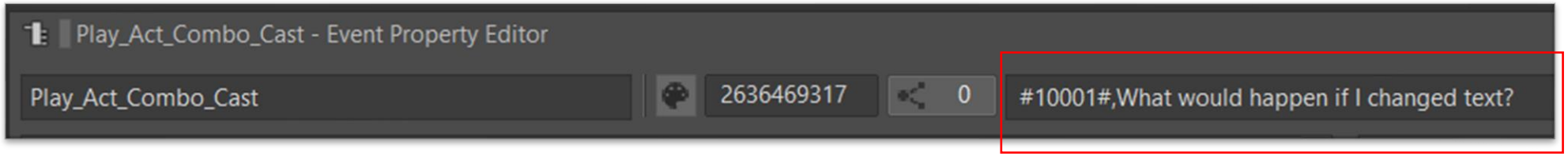
Other Functions&Tips – How to update Event Notes with one click?

The only step:

In the data table of the plugin's main window, enter the new Notes in the cell in the Notes column. On the changed Notes cell, right-click the mouse, and in the pop-up menu options, click “Renew Notes". You can select multiple rows to process them in batches at once.

If the Event name is correct, the Notes of the corresponding Event in the Wwise project will be updated immediately.

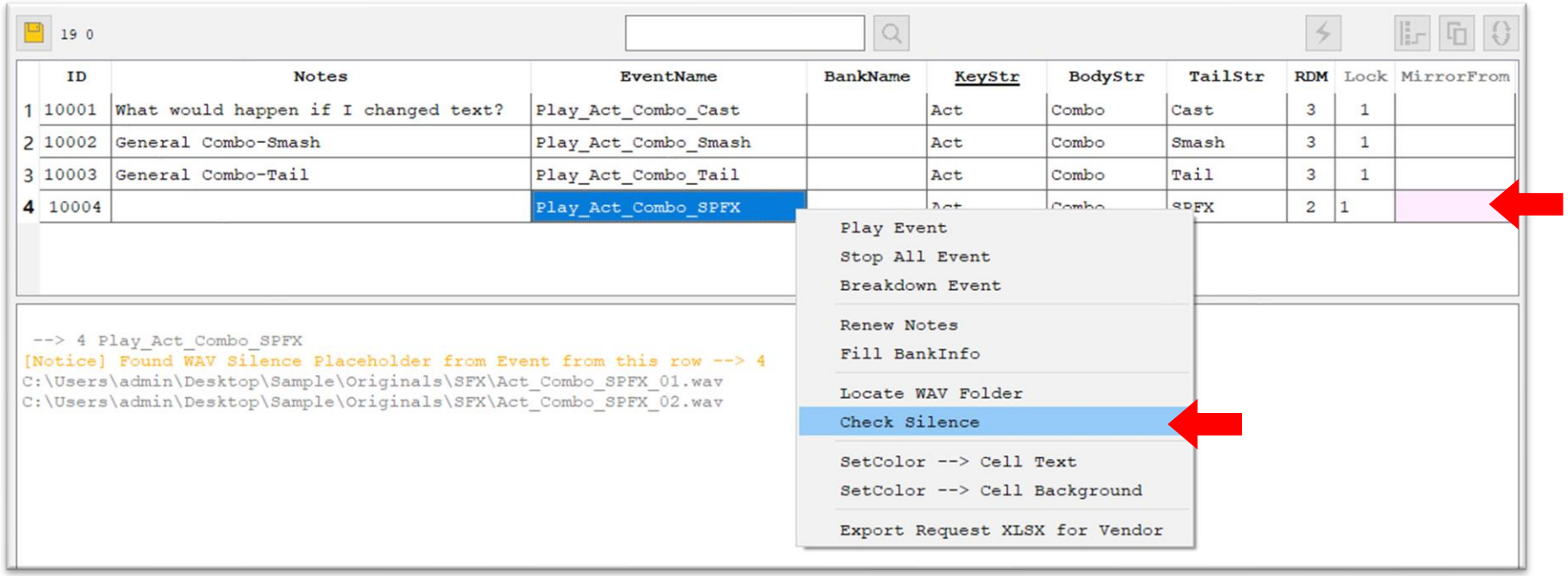




Other Functions&Tips – How to check if all silence placeholders have been replaced with official resources and if there are any omissions?

Select the rows that need to be confirmed, select any one or more cells, right-click the mouse, and in the menu options, click "Check Silence".

If the cell background color in the MirrorFrom column of the current row turns **PINK**, it means that there are still **silence placeholders** that have not been overwritten or replaced for the WAV related to this Event (the log console will also display specific information); On the contrary, if the cell in the MirrorFrom column does not change color, it means that all the WAVs under this Event are no longer silence placeholders.

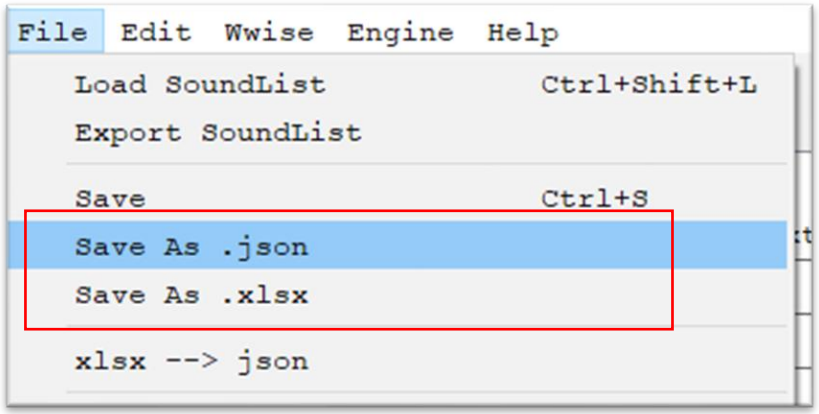


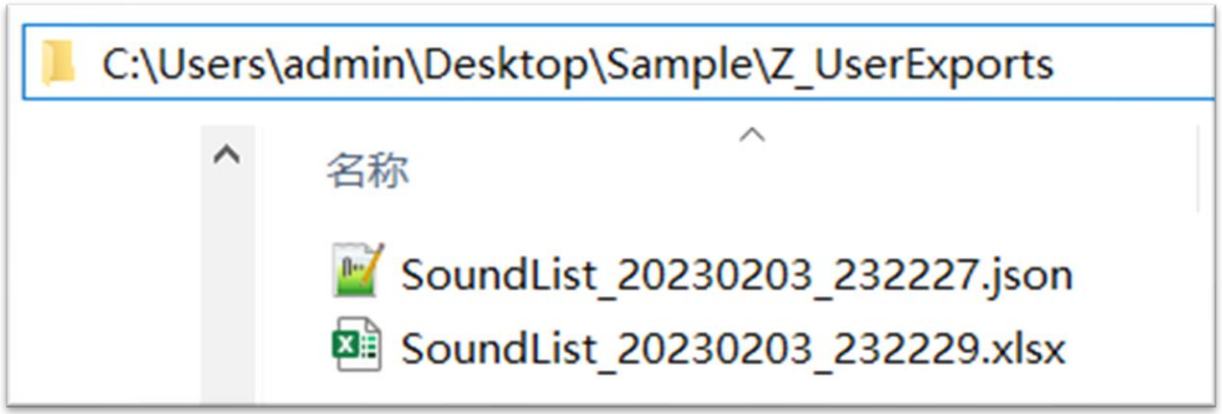
Other Functions&Tips – Save as JSON, xlsx

By clicking "File→Save As .json" in the menu bar, you can export a .json file copy of all the data in the current table and save it in the "Z_UserExports" folder under the current Wwise project path.

By clicking "File→Save As .xlsx" in the menu bar, you can export an .xlsx file copy of all the data in the current table and save it in the "Z_UserExports" folder under the current Wwise project path.

Note: The file name of the saved file is a concatenation of "SoundList" and the current time information "year month day hour minute second". Therefore, we can determine the initial generation time of the file from the file name.





Note: The "Z_UserExports" folder under the Wwise project path is automatically generated by the tool and is used to store temporary data files such as JSON and xlsx saved by the user.

Other Functions&Tips – xlsx → JSON

We can convert an xlsx audio table that meets the format requirements of the plugin into a JSON that can be directly loaded by the plugin as a table, including the **text in the cells, text color, and cell background color**.

First, we click "File→xlsx --> json" in the menu bar and select the xlsx that needs to be converted. If the xlsx is a valid xlsx, the converted JSON file will be saved in the "Z_UserExports" folder under the current Wwise project path. The naming of the JSON file is a concatenation of "SoundList" and the current time information "year month day hour minute second".

Format requirements for the xlsx table:
1) The text and order in the cells of the first row must be exactly the same as the table header in the plugin's table.
2) The actual data content starts from the second row and is recorded row by row.

	A	B	C	D	E	F	G	H	I	J
1	ID	Notes	EventName	BankName	KeyStr	BodyStr	TailStr	RDM	Lock	MirrorFrom
2	10005	Wall-E	Play_UI_AA		UI	AA		1	1	
3	10006	Pikachu	Play_Act_BB		Act	BB		2	1	
4	10007	Optimus Prime	Play_FT_CC		FT	CC		3	1	
5	10008	Sailor Moon	Play_Amb_DD		Amb	DD		4	1	
6	10009	Naruto	Play_UI_EE		UI	EE		2	1	
7	10010	Detective Conan	Play_Act_FF		Act	FF		2	1	
8	10011	Peter Parker	Play_FT_GG		FT	GG		3	1	
9	10012	Tony Stark	Play_Amb_HH		Amb	HH		4	1	
10	10013	Naruto Sasuke	Play_UI_II		UI	II		3	1	
11	10014	Hulk	Play_Act_JJ		Act	JJ		2	1	
12	10015	Jack Sparrow	Play_FT_KK		FT	KK		3	1	

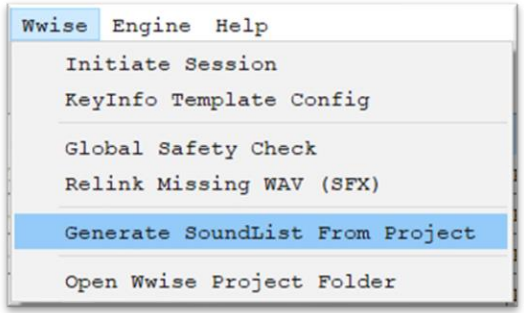

```
"$ProjectStr$": "Sample.wproj",
"Data_SoundList": {
    "10005": {
        "ID_textColor": "#000000",
        "ID_bgColor": "#ffffff",
        "Notes": {
            "text": "Wall-E",
            "textColor": "#000000",
            "bgColor": "#ffffff"
        },
        "EventName": {
            "text": "Play_UI_AA",
            "textColor": "#000000",
            "bgColor": "#ffffff"
        },
        "BankName": {
            "text": "",
            "textColor": "#000000",
            "bgColor": "#ffffff"
        },
        "KeyStr": {
            "text": "UI",
            "textColor": "#000000",
            "bgColor": "#ffffff"
        },
        "BodyStr": {
            "text": "AA",
            "textColor": "#000000",
            "bgColor": "#ffffff"
```

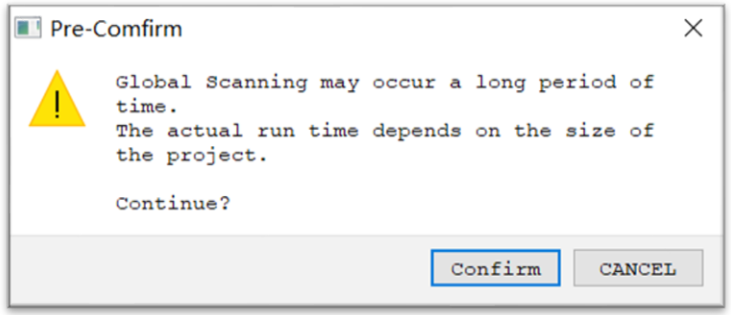
Other Functions&Tips – Generate JSON based on Project

If we are using the plugin for the first time, the info.json file initially generated by the plugin itself does not record any useful data. If at this time, there is already a large amount of data in Wwise, and we want the plugin to take over Wwise, we need to first scan all the Events inside Wwise to generate a usable JSON file.

Step 1: Click "Wwise→Generate SoundList from Project" in the menu bar. At this point, a confirmation window will pop up. If there are no issues, click confirm and wait for execution.

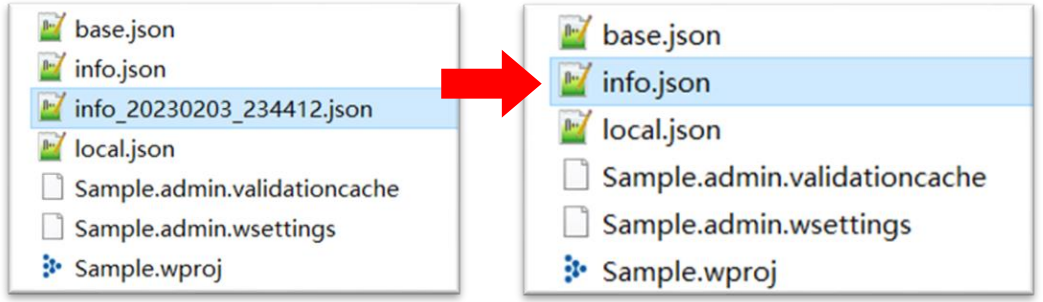
Note: If there are a large number of Events in the Wwise project, this process may cause the plugin window to "freeze". In this case, do not do anything and wait until the execution is completed and the window returns to normal.





Step 2: The newly generated JSON file will directly appear in **the root directory of the Wwise project**, with a naming format that concatenates "info" and "time information".

At this point, **manually delete the old blank info.json file**, directly **manually change the naming of this newly generated JSON file to "info.json"**, and then **restart the plugin**.

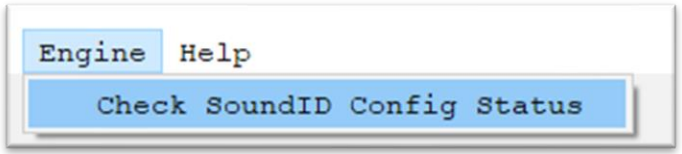


Other Functions&Tips – Check audio ID mounting status

If the engine side can collect and statistics the configuration locations of all global Sound IDs and compare them with the audio table, and save the differences after comparison to a JSON file in the format shown in the figure below.

```
{
    "ID_ExistInSoundList_NotExistInEngine": ["ID1"],
    "ID_ExistInEngine_NotExistInSoundList": ["ID2"],
    "ID_FoundDuplicate_InEngine": {
        "ID3": [
            "location1",
            "location2"
        ]
    },
    "Event_ExistInSoundList_NotExistInEngine": ["Play_01"],
    "Event_ExistInEngine_NotExistInSoundList": ["Play_02"],
    "Event_FoundDuplicate_InEngine": {
        "Play_03": [
            "location1",
            "location2"
        ]
    }
}
```

We can then click "Engine→Check SoundID Config Status" in the menu bar, to display the comparison report in the log console and view the actual mounting status of the audio IDs.

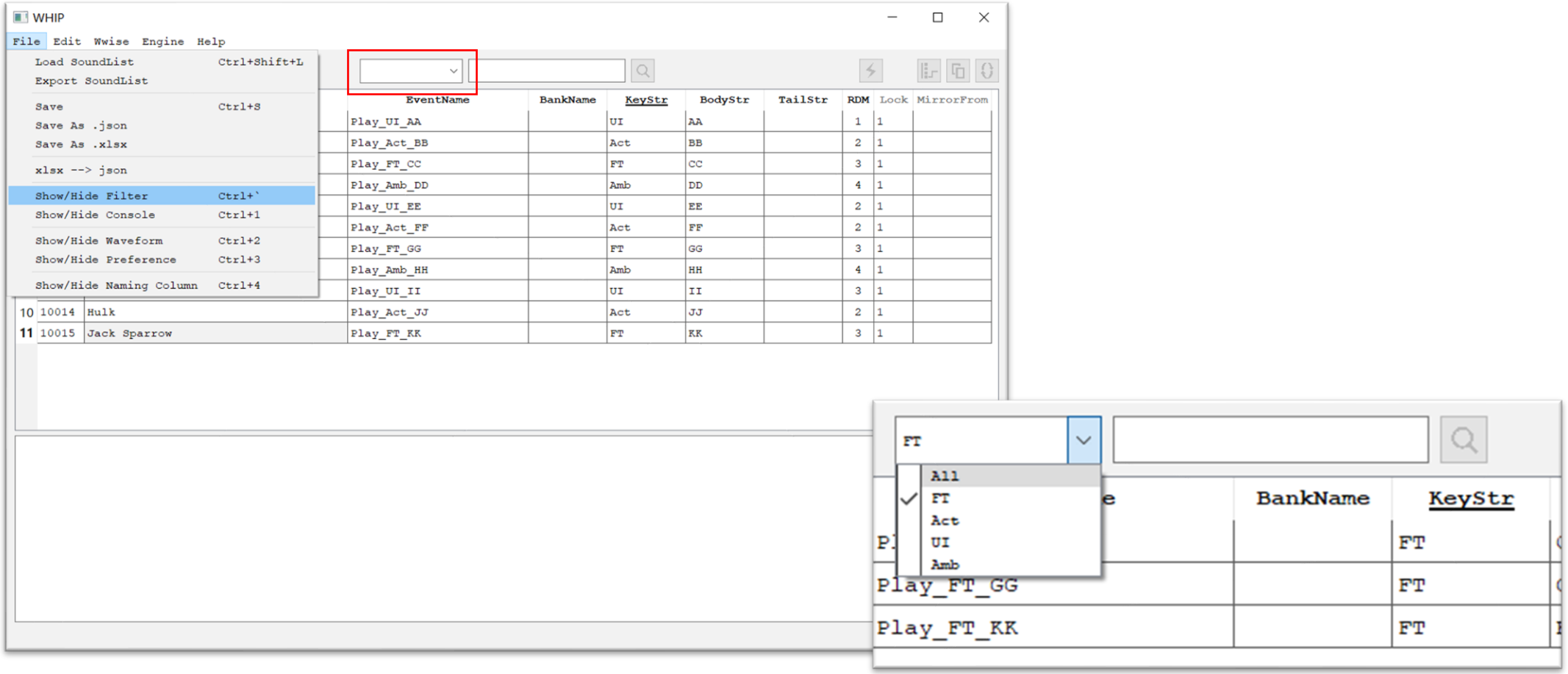



```
>>>>>>>>>>>>>>>>>>>> SoundID Config Status Report <<<<<<<<<<<<<<<<<<<<
[Warning] Found ID that exist in SoundList, but not exist in Engine!
 --> ID1
[Warning] Found ID that exist in Engine, but not exist in SoundList!
 --> ID2
[Warning] Found Same ID that exist in MULTIPLE PLACES in Engine!
{
    "ID3": [
        "location1",
        "location2"
    ]
}
[Warning] Found Event that exist in SoundList, but not exist in Engine!
 --> Play_01
[Warning] Found Event that exist in Engine, but not exist in SoundList!
 --> Play_02
[Warning] Found Same Event that exist in MULTIPLE PLACES in Engine!
{
    "Play_03": [
        "location1",
        "location2"
    ]
}
```

Other Functions&Tips – Filter

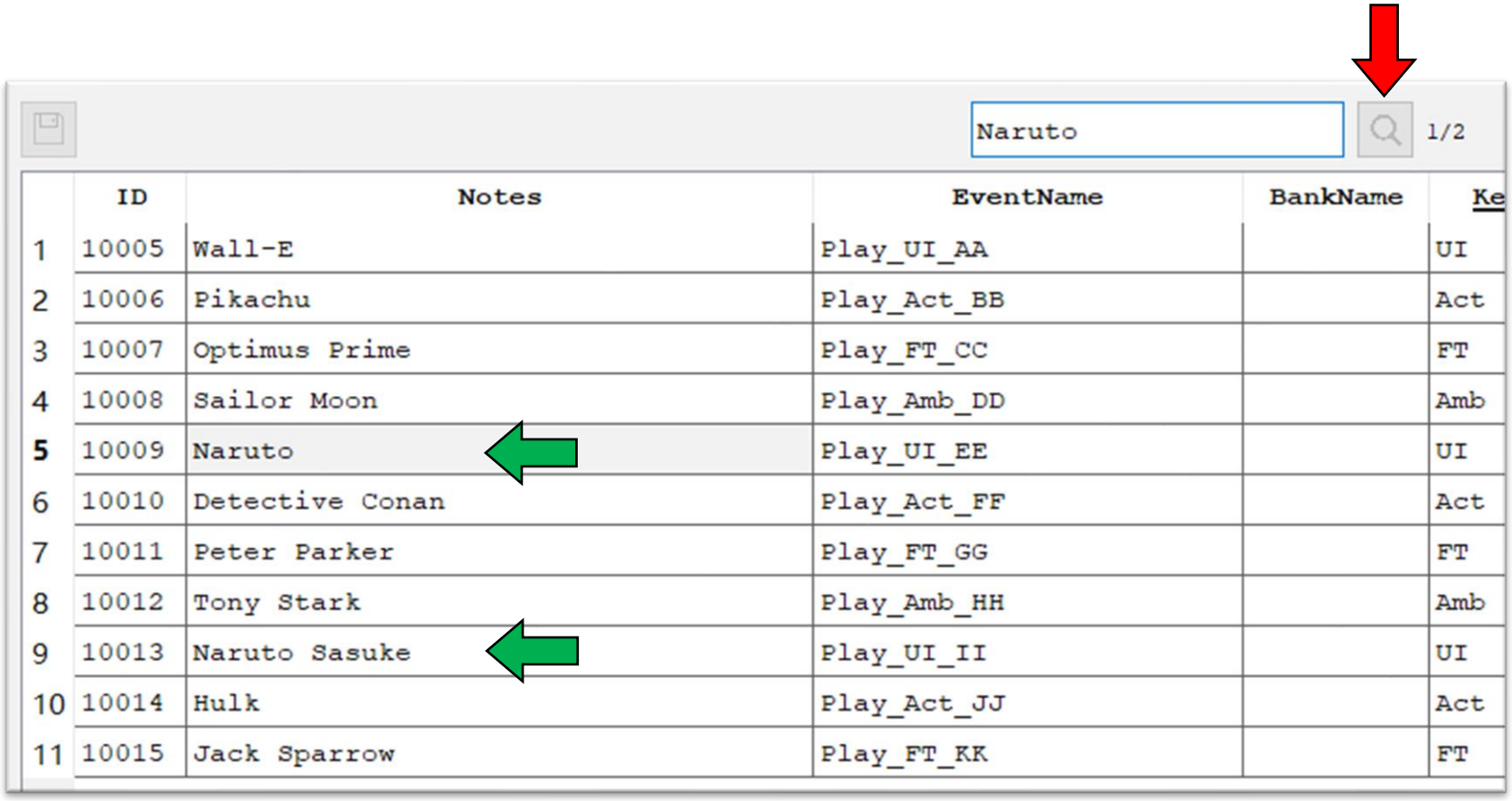
【Use "Filter" to display data by category】

When there is a large amount of data in the table, we can click "File→Show/Hide Filter" in the menu bar, and the filter will appear at the top of the window. Designers can click one or several categories to display only the selected relevant content and hide irrelevant content.



Other Functions&Tips – Search text

Enter the text to be searched in the text box at the top of the main interface, click the search button, and the cells containing the target text will be highlighted and displayed in the table. If there are multiple results, clicking the search button continuously will automatically switch between the multiple results.

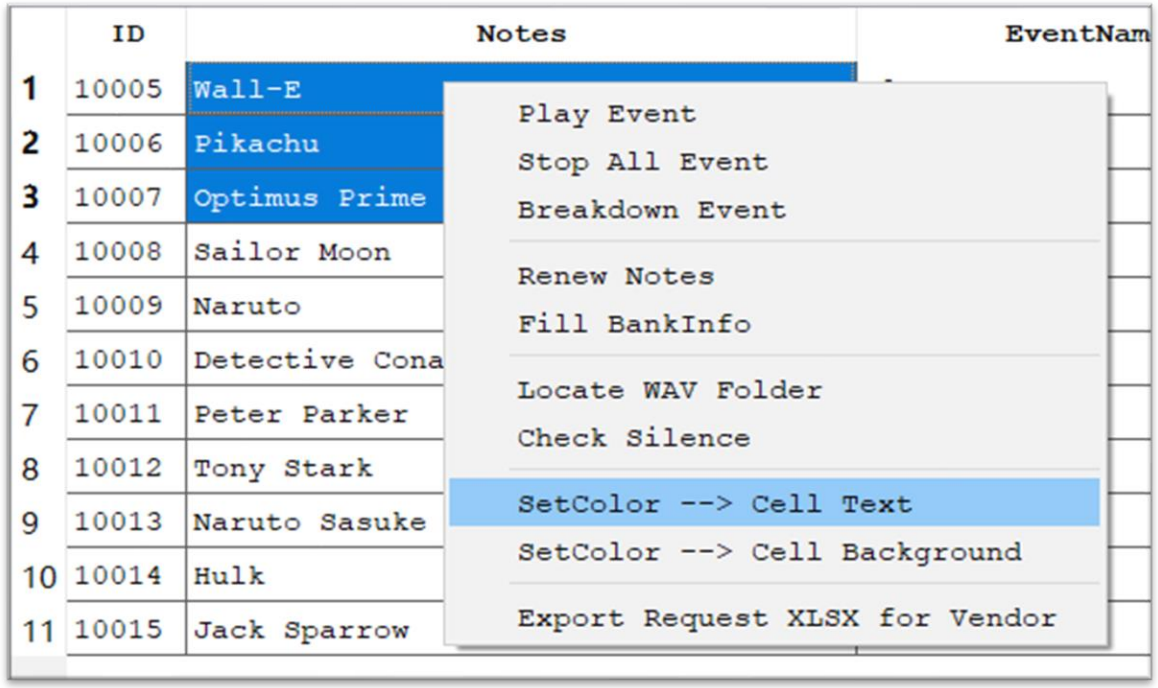


	ID	Notes	EventName	BankName	Ke
1	10005	Wall-E	Play_UI_AA		UI
2	10006	Pikachu	Play_Act_BB		Act
3	10007	Optimus Prime	Play_FT_CC		FT
4	10008	Sailor Moon	Play_Amb_DD		Amb
5	10009	Naruto	Play_UI_EE		UI
6	10010	Detective Conan	Play_Act_FF		Act
7	10011	Peter Parker	Play_FT_GG		FT
8	10012	Tony Stark	Play_Amb_HH		Amb
9	10013	Naruto Sasuke	Play_UI_II		UI
10	10014	Hulk	Play_Act_JJ		Act
11	10015	Jack Sparrow	Play_FT_KK		FT

Other Functions&Tips – Set text color and cell background color

Simply select the cells to be edited, right-click "SetColor --> Cell Text" or "SetColor --> Cell Background", select the color, and confirm to take effect.

Note: The color information will be saved in info.json along with the text information.

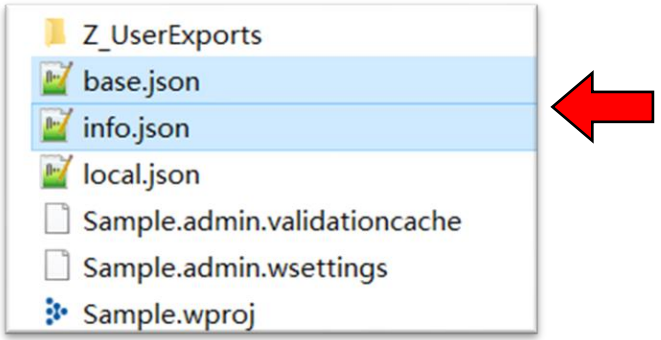


Other Functions&Tips – Version Control

The plugin will generate some files during operation.

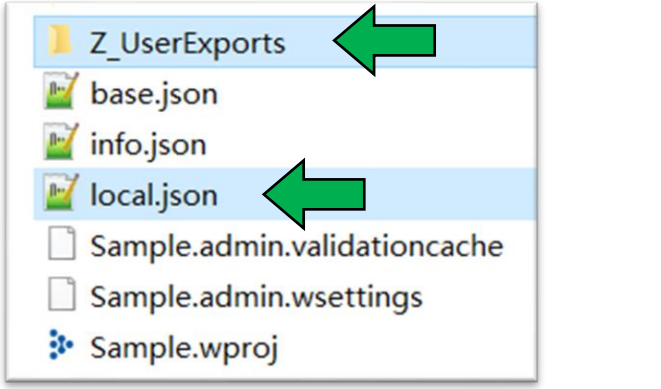
The files that need to participate in version control (Perforce, UGIT, SVN, etc.), be uploaded and downloaded along with the Wwise project data, participate in multi-person collaboration synchronization, difference comparison, and conflict resolution include:

1) The 2 JSON files in the root directory of the Wwise project: **"info.json"** and **"base.json"**
2) The **"SoundList.json"** exported to the engine path for client-side reading



The files that do not need to participate in version control and are recommended to be added to the **ignore list** include:

1) **"local.json"** in the root directory of the Wwise project, and 1 folder generated by the plugin: **"Z_UserExports"**
2) All files in the plugin's own directory

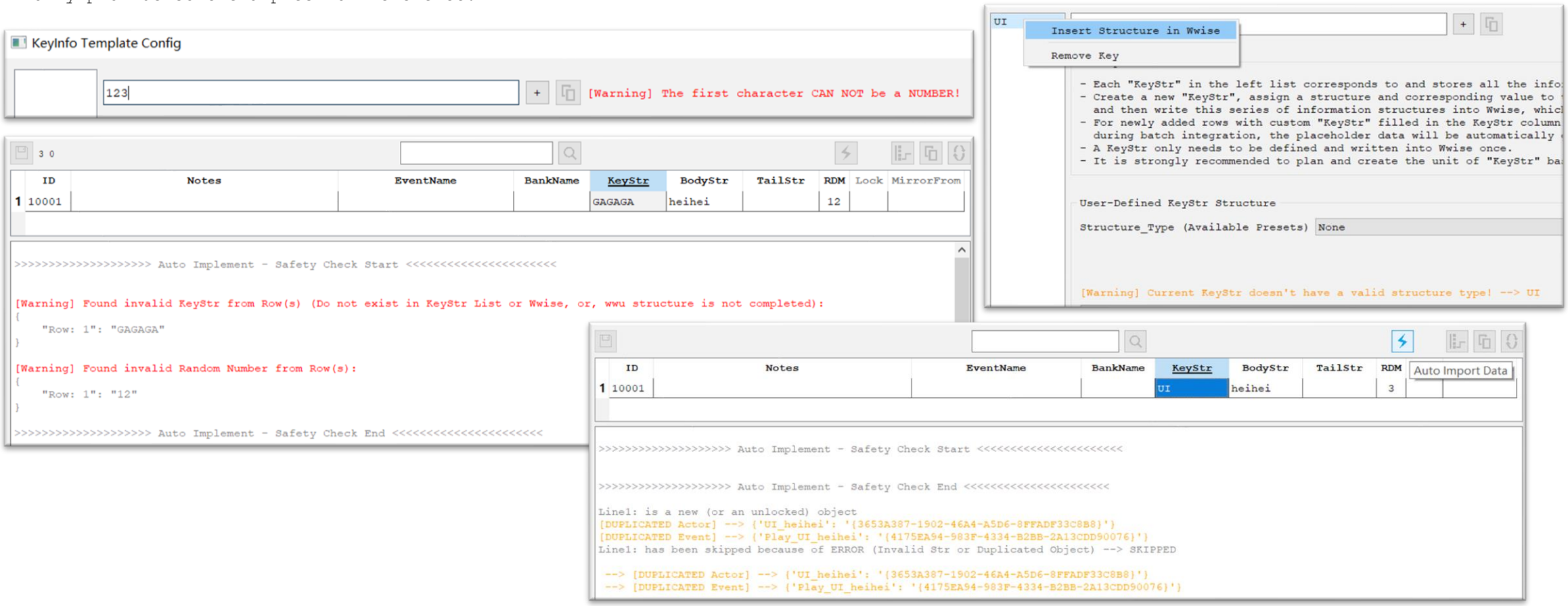


Other Functions&Tips - Error Description

When the plugin is running normally, we can see many normal and friendly logs and reports in the log console. But when the plugin runs abnormally or detects that the user may have made an operation mistake, we may also see various **safety prompts** generated by the plugin.

To assist designers in using the plugin more smoothly, the developer has arranged as many safety prompts as possible in every corner. **The highlighted font colors, warning windows**, and other information can basically directly attract the attention of designers, so that they can quickly notice some potential operation safety issues.

Since there are a lot of safety prompt information set in the plugin, the manual will not elaborate on them one by one for now, and only provide some examples for reference:



Developer's Message

"Surrounded by invisible chores, unable to concentrate on creation"
In recent years, more and more designer friends around me have started to discuss this troublesome issue.

Different companies have different manpower and financial resources, and the work compensation brought by the external environment, even for Wwise which is already trying to turn the situation around, may be helpless. Moreover, the work tasks, stress levels, and comfort of designers also largely depend on the actual operation and management of the project.

Managers who focus on "managing affairs" are more concerned about whether designers can "adapt to changes", "complete tasks on time", and "solve hidden dangers" when completing project requirements.

Managers who focus on "managing data" are more concerned about whether the large amount of data produced by designers is "complete and organized", "clear and concise", "easy to maintain", and "easy to handover".

Managers who focus on "achieving performance" are more concerned about whether designers can create business "highlights" or whether they can solve the urgent needs of superiors in a timely manner.

The managers that designers face may be one of the above, or a combination of several. Different corporate cultures, systems, and organizational structures have different requirements for managers, and thus, managers' attitudes towards designers naturally vary "in hundreds of ways". Accordingly, the work focus, value orientation, right and wrong, success and failure, and honor and disgrace of different designers also differ greatly.

So, how many designers can achieve success on both sides? Or, how many designers can still design "excellent works" while achieving success on both sides? In reality, not everyone can make choices.

If designers are always troubled by invisible chores and find it difficult to focus on truly meaningful creation, it will not be a long-term solution for individuals, teams, or the industry.

Wwise has thoroughly summarized and sorted out all aspects of the game audio business, providing designers with a broad operational perspective. Then "WHIP" continues to take the baton, throwing out a brick to attract jade, and discusses with designer friends: how to take back all the time that was stolen from us in daily life!

If we are always busy running around, we will miss too many interesting sceneries in life, won't we?