

JAudio
Modding
TOOLkit



xayrga

JAMToolsDocumentation

Contents

Contents 2

About JAMTools (JAudioModdingToolkit)	4
Terminology.....	4
IBNK	4
WSYS.....	4
BMS	4
AAF	4
BAA	4
SFT / BSFT	5
Sound Table / ST / BST	5
SC/BSC	5
JAudio version table	6
Installing JAMTools into your game	7
About the Audio_Modding folder	13
Jampacked 14	
Command Line	14
unpacking.....	14
Packing.....	14
The PROJECT.JSON file	15
originalFile	15
projectName	15
format.....	15



xayrga

JAMToolsDocumentation

Includes.....	16
Formats.....	18
AAF	18
BAA	19
Notes	20
Hashes	20
WSYSBUILDER.....	22
About.....	23
Command Line	23
Project Output	24
manifest.json	24
wavetable.json.....	25
Scenes.....	28
The folders and their function.....	29
WAV	29
REF	29
Custom.....	29
Where do I find .ws / .wsys files?	29
BAST 30	
About.....	31
Command Line	31
Sound ID's.....	31
JASE	32
REBUILDING JASE	34



JAMToolsDocumentation

About JAMTools (JAudioModdingToolkit)

JAMTools is a multi-game toolset for modifying the audio systems in various Nintendo Gamecube titles. It is designed for ease of use, and conversion of the binary data into a human readable and human editable format.

Terminology

IBNK

IBNK is the “Instrument Bank” format. This contains information such as the pitching + grouping of sounds, the fadein and fadeout (attack / release) , and oscillation.

WSYS

The WSYS is the “WaveSystem”. The wavesystem controls when and how sounds are loaded, where they are loaded from, and their allocation in memory.

BMS

BMS is a “Binary Musical Sequence”. It contains music bytecode that tells the sequence engine how to play notes.

AAF

AAF is the Audio Archive File. It contains various sections, like the IBNK / WSYS / sometimes BMS / Sequence Table / BARC header. It is the file that points to all of the initialization data for the audio system.

BAA

BAA is the Binary Audio Archive. It is an updated version of the Audio Archive File, with a unique quirk of being able to contain more of itself. (Though that functionality is only ever seen in one game)



JAMToolsDocumentation

SFT / BSFT

The SFT is the “Stream File Table” or the “Binary Stream File Table”. It contains file paths to match up the files on the disk to entries in the sound table

Sound Table / ST / BST

The “Sound Table” or in newer games “Binary Sound Table” is the container that contains all parameters for the sound playback. The sound table is split up into categories that determine where the sounds are played from, and what volume / source they use (as some entire categories can have their effects and playback adjusted via code at any time in the game, for example, when you pause in windwaker)

SC/BSC

Sequence Collection or Binary Sequence Collection. Contains the information to find the BMS files usually, or sometimes substitutes for the “SE.BMS” on newer games.



JAMToolsDocumentation

JAudio version table

Pikmin 1	Hybrid	V1 banks, v1 sequences, custom Audio Archive format.
Luigi's Mansion	v1	Fully V1 system
Super Mario Sunshine	v1	Fully V1 system
Pikmin 2	v1*	Fully V1 system with custom streamtable format.
Windwaker	v1	Fully V1 system
Mario Kart Double Dash	Hybrid	v1 Banks, v2 Archive (BAA) , v2 Sequence format, v2 soundtable with v1 entries, nested BAA's (BAAC)
Four Swords Adventure	Hybrid	V1 and V2 banks + v2 system (weird BST)
Donkey Kong Jungle Beat	v2	Fully V2 system.
Twilight Princess	V2	Fully V2 system.
Super Mario Galaxy 1	V2	Fully V2 system + special subsystem
Super Mario Galaxy 2	V2	Fully V2 system + special subsystem
Links Crossbow Training	V2	Fully v2 system
Pacman VS	V2	Fully v2 system.
Zelda Collectors Edition	V2	Fully V2 system.



JAMToolsDocumentation

Installing JAMTools into your game

1. Visit <https://www.xayr.ga/tools/SoundModdingToolkit/> and select the most recent version of the tool.

Index of /tools/SoundModdingToolkit

Name	Last modified	Size	Description
Parent Directory	-	-	-
3.1.0/	2020-11-20 18:16	-	-
3.1.2/	2020-11-22 23:05	-	-
3.1.3/	2020-12-22 19:05	-	-
3.1.4/	2020-11-24 10:53	-	-
3.1.5/	2020-11-24 16:32	-	-
3.1.6/	2020-11-29 01:20	-	-
3.1.7/	2020-12-04 00:14	-	-
4.0.0/	2020-12-20 21:52	-	-
4.0.1/	2020-12-22 19:05	-	-
4.0.3/	2021-01-15 19:37	-	-
4.1.0/	2021-02-06 17:07	-	-
4.1.1/	2021-02-10 19:23	-	-
4.1.3/	2021-03-20 16:05	-	-
4.1.5/	2021-05-02 17:46	-	-
5.0.2/	2021-08-15 15:39	-	-
5.2.2/	2021-09-06 02:07	-	-

Apache/2.4.29 (Ubuntu) Server at www.xayr.ga Port 80

2. Select the specific tooling for the game you are modifying.

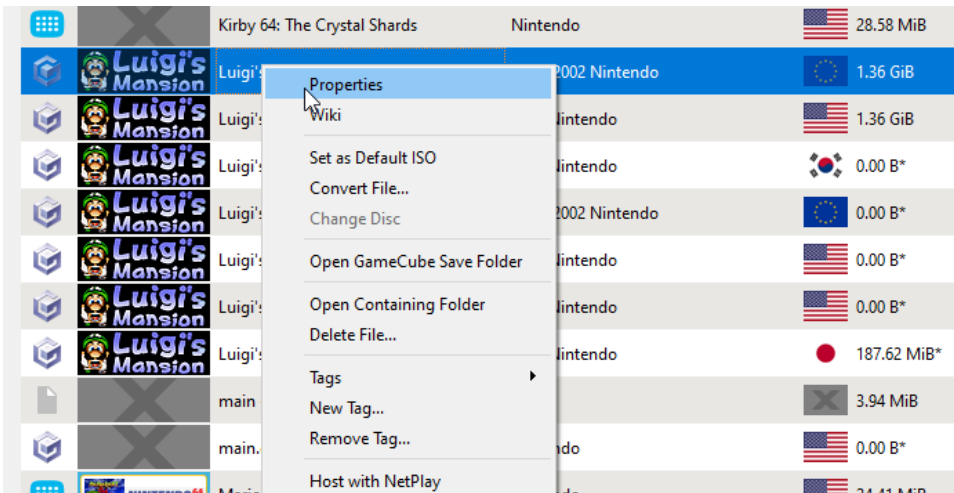
Name	Last modified	Size	Description
Parent Directory	-	-	-
DOUBLEDASH_SoundModdingToolkit_5.2.2.zip	2021-09-06 02:06	2.0M	-
DonkeyKongJungleBeat_SoundModdingToolkit_5.2.2.zip	2021-09-06 02:06	2.0M	-
FourSwordsAdventure_SoundModdingToolkit_5.2.2.zip	2021-09-06 02:06	2.0M	-
LinksCrossnbowTraining_SoundModdingToolkit_5.2.2.zip	2021-09-06 02:06	2.0M	-
LuigisMansion_SoundModdingToolkit_5.2.2.zip	2021-09-06 02:06	2.0M	-
PIKMIN2_SoundModdingToolkit_5.2.2.zip	2021-09-06 02:06	2.0M	-
PacmanVS_SoundModdingToolkit_5.2.2.zip	2021-09-06 02:06	2.0M	-
Pikmin1_SoundModdingToolkit_5.2.2.zip	2021-09-06 02:06	2.0M	-
Sunshine_SoundModdingToolkit_5.2.2.zip	2021-09-06 02:06	2.0M	-
SuperMarioGalaxy2_SoundModdingToolkit_5.2.2.zip	2021-09-06 02:06	2.0M	-
SuperMarioGalaxy_SoundModdingToolkit_5.2.2.zip	2021-09-06 02:06	2.0M	-
TwilightPrincess_SoundModdingToolkit_5.2.2.zip	2021-09-06 02:06	2.0M	-
WINDWAKER_SoundModdingToolkit_5.2.2.zip	2021-09-06 02:06	2.0M	-
ZeldaCollectorsEdition_SoundModdingToolkit_5.2.2.zip	2021-09-06 02:06	2.0M	-

Apache/2.4.29 (Ubuntu) Server at www.xayr.ga Port 80

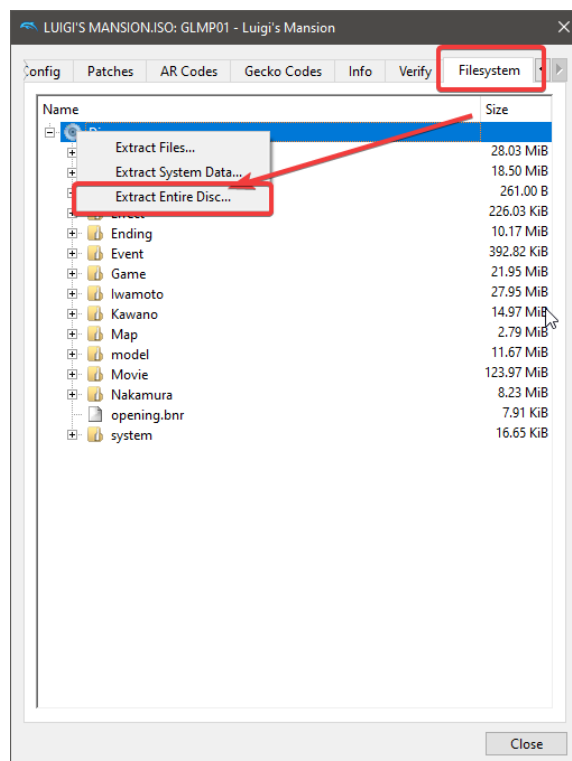


JAMToolsDocumentation

3. You now have to extract your ROM. Open Dolphin and select “Properties” on your game.

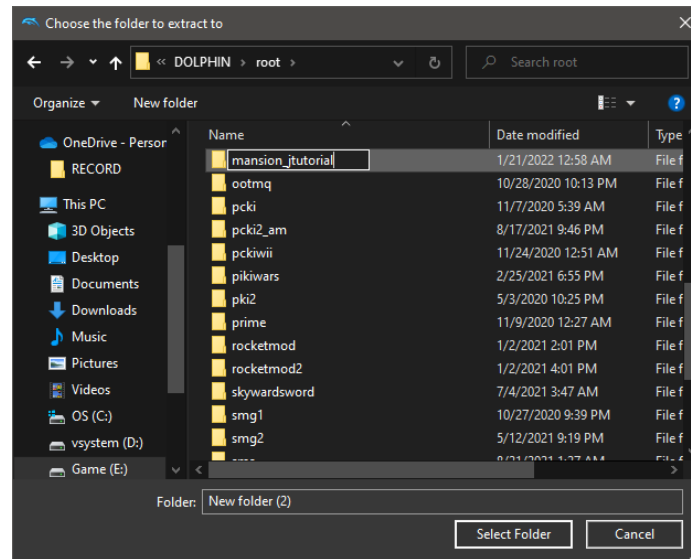


4. On the properties of this game, select “Filesystem”. Right click on disk then select “Extract Entire Disc”

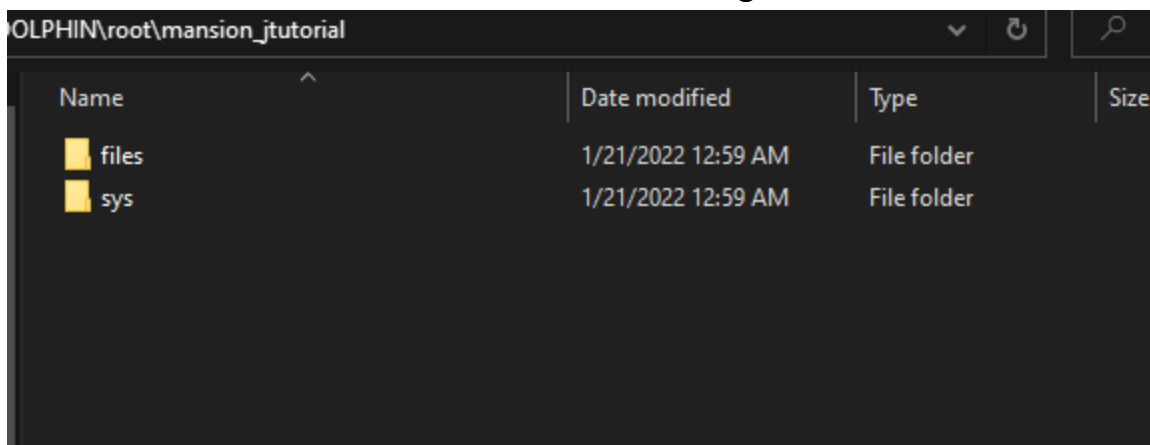


JAMToolsDocumentation

5. Create a new folder, and extract the contents into it by pressing “Select Folder”

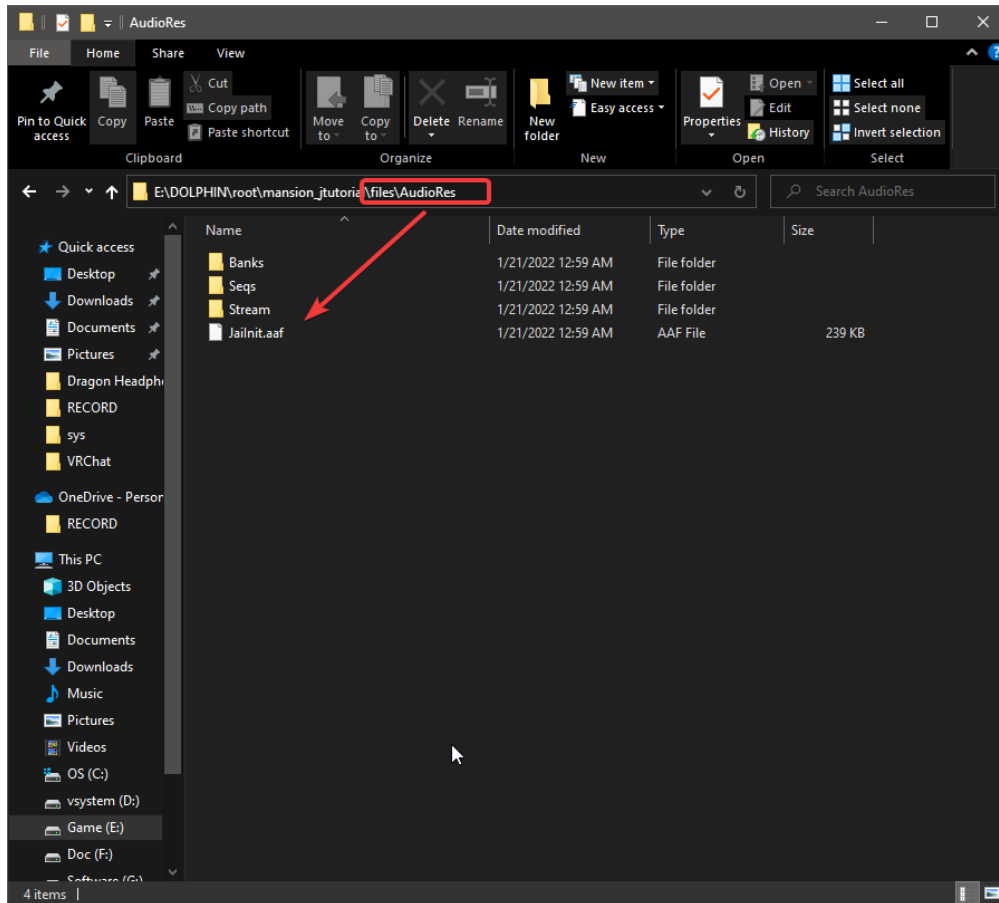


Inside of that folder should look something like this.

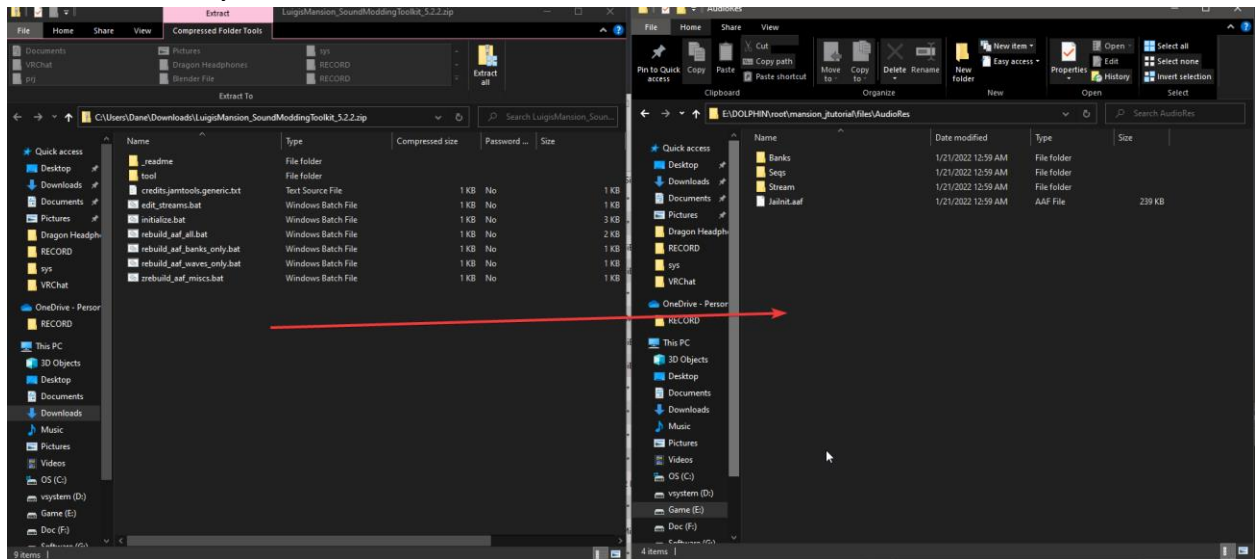


JAMToolsDocumentation

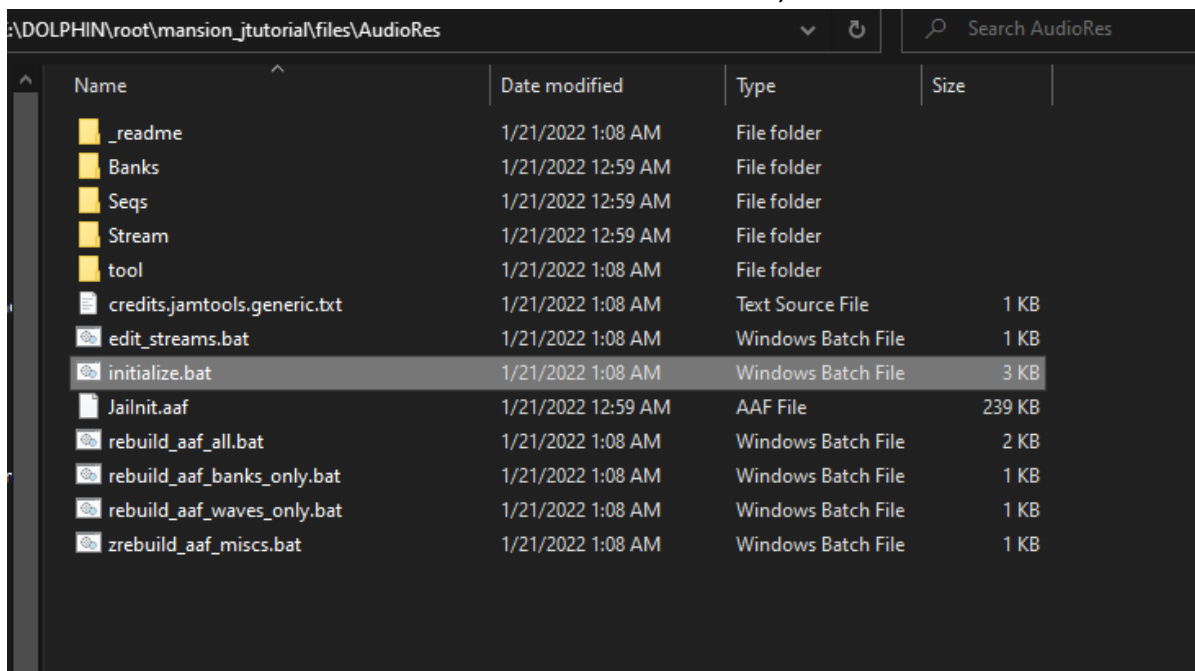
6. Open the “Files” folder. And locate the AudioRes folder



- Open the ZIP file you downloaded for your game, then copy all of the contents into your AudioRes folder.



- Your folder should now look like this. When it does, run the “Initialize.bat”



JAMToolsDocumentation

9. Wait for extraction to complete. Then the toolkit is successfully installed. You should see an “Audio_Modding” folder.

```
Unpacking AAF....
"E:\DOLPHIN\root\mansion_jtutorial\Files\AudioRes\Banks"
Jamped JAudio Archive packer / unpacker
Unpack AAF
Unpacking clusters first...
-> Wrote 0.bnk
-> Wrote 1.bnk
-> Wrote 2.bnk
-> Wrote 3.bnk
-> Wrote 4.bnk
-> Wrote 5.bnk
-> Wrote 0.wsy
-> Wrote 1.wsy
-> Wrote 2.wsy
-> Wrote 3.wsy
Unpacking sections....
-> Wrote 0.bst
-> Wrote 1.arc
-> Wrote 2.stm
-> Wrote 3.dat
-> Wrote 4.dat
Done.
wsystool JAudio WSYS packer / unpacker
->Extract (LuiSec0_0.aw) [#####] (119/119)
Transforming ADPCM data... (may take a while)
->Transform (LuiSec0_0.aw) [#####] (119/119)
Writing wavetable...
Writing Project File
wsystool JAudio WSYS packer / unpacker
->Extract (LuiSec2_0.aw) [#####] (1/1)
Transforming ADPCM data... (may take a while)
->Transform (LuiSec2_0.aw) [#####] (1/1)
Writing wavetable...
Writing Project File
wsystool JAudio WSYS packer / unpacker
->Extract (LuiSec1_0.aw) [#####] (558/558)
Transforming ADPCM data... (may take a while)
->Transform (LuiSec1_0.aw) [#####] (558/558)
Writing wavetable...
Writing Project File
wsystool JAudio WSYS packer / unpacker
->Extract (LuiSec2_0.aw) [#####] (348/348)
Transforming ADPCM data... (may take a while)
->Transform (LuiSec2_0.aw) [#####] (57/348)
```

Name	Date modified	Type	Size
backup	1/21/2022 1:09 AM	File folder	
_readme	1/21/2022 1:08 AM	File folder	
Audio_Modding	1/21/2022 1:09 AM	File folder	
Banks	1/21/2022 12:59 AM	File folder	
Seqs	1/21/2022 12:59 AM	File folder	
Stream	1/21/2022 12:59 AM	File folder	
tool	1/21/2022 1:08 AM	File folder	
credits.jamtools.generic.txt	1/21/2022 1:08 AM	Text Source File	1 KB
edit_streams.bat	1/21/2022 1:08 AM	Windows Batch File	1 KB
initialize.bat	1/21/2022 1:08 AM	Windows Batch File	3 KB
Jailnit.aaf	1/21/2022 12:59 AM	AAF File	239 KB
rebuild_aaf_all.bat	1/21/2022 1:08 AM	Windows Batch File	2 KB
rebuild_aaf_banks_only.bat	1/21/2022 1:08 AM	Windows Batch File	1 KB
rebuild_aaf_waves_only.bat	1/21/2022 1:08 AM	Windows Batch File	1 KB
zrebuild_aaf_miscs.bat	1/21/2022 1:08 AM	Windows Batch File	1 KB



JAMToolsDocumentation

About the Audio_Modding folder

The Audio_Modding folder is generated by the tool called “jampacked”. The various other folders within it are created by various different tools (ibnkttool, wsysbuilder, barctool, etc....) . The “project.json” within it contains the information required to rebuild the AAF / BAA.

Name	Date modified	Type	Size
IBNK0	1/21/2022 1:09 AM	File folder	
IBNK1	1/21/2022 1:09 AM	File folder	
IBNK2	1/21/2022 1:09 AM	File folder	
IBNK3	1/21/2022 1:09 AM	File folder	
IBNK4	1/21/2022 1:09 AM	File folder	
IBNK5	1/21/2022 1:09 AM	File folder	
include	1/21/2022 1:09 AM	File folder	
Sequences	1/21/2022 1:09 AM	File folder	
SoundTable	1/21/2022 1:09 AM	File folder	
WS0	1/21/2022 1:09 AM	File folder	
WS1	1/21/2022 1:09 AM	File folder	
WS2	1/21/2022 1:09 AM	File folder	
WS3	1/21/2022 1:09 AM	File folder	
project.json	1/21/2022 1:16 AM	JSON Source File	3 KB

The “include” folder will contain the various binary forms of the AAF / BAA sections. The project.json will have references to each of these sections when it is generated. If you would like information on editing the AAF or BAA, look for the tool “jampacked” in the documentation.

WS*** folders usually contain folders generated by wsysbuilder. (Wave system data)

IBNK*** folders usually contain contents generated by ibnkttool (Instrument Bank Data)

Sequences usually contains the output of barctool.

SoundTable is the work of bast.

While in some games (such as pikmin 1) these folders may be named differently, it will be easy to understand the function of a particular section.



Jampacked

JAMPACKED



Jampacked is a tool used for packing / unpacking the Audio Archive File or the Binary Audio Archive.

Command Line

unpacking

Jampacked.exe <unpack> <aaf/baa/bx file> <output_folder>

Unpacks the AAF / BAA or BX file to a project.json + includes.

Packing

Jampacked.exe <pack> <project_folder> <output_file>

Packs the project folder back into its source format, or the format specified by the project.json



xayrga

JAMToolsDocumentation

The PROJECT.JSON file

The project.json file contains the information needed to rebuild your audio archive. It is designed to be edited, but for jumpacked it comes in two variations.

The base header will always contain “originalFile” “projectName” and “format”

```
E: > DOLPHIN > root > mansion_jtutorial > files > AudioRes > Audio_Modding > {} pr
1  {
2    "originalFile": "JaiInit.aaf",
3    "projectName": "Audio_Modding",
4    "format": "AAF",
5    "banks": [
6      {
```

originalFile

the name of the original audio archive, used if a project output file is not specified.

projectName

this is the name of the project folder, it is not needed either.

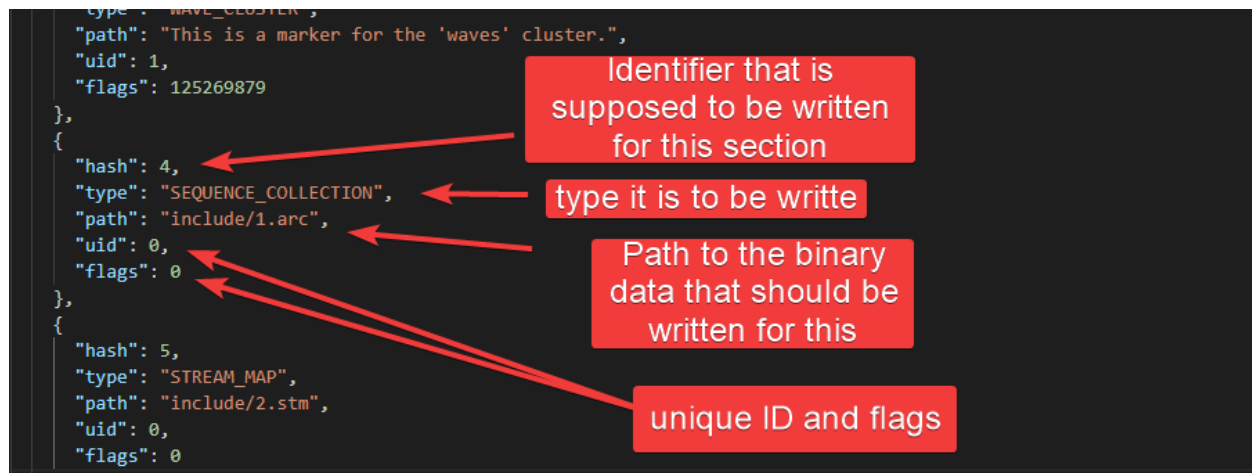
format

Format is the important field that specifies what information it should be looking for during the rebuild. It will also determine the output format of the project when it is rebuilt.



JAMToolsDocumentation

Includes



Includes dictate the sections that are loaded into the audio archive.

When extracted, the includes folder includes any sub-files that were contained inside of the AAF or BAA, such as any WSYS, IBNK, the Stream Table, any other BAA's or AAF's (yes, they can be nested, have fun.)

Inside of the includes folder are the various components of the BAA or AAF.

PC > Doc (F:) > TOOLSET > JTK > jampacked > jampackednf > bin > Debug > out > include

Name	Date modified	Type	Size
0.bnk	9/5/2021 7:14 PM	BNK File	3 KB
0.bst	9/5/2021 7:14 PM	BST File	2 KB
0.wsy	9/5/2021 7:14 PM	WSY File	3 KB
1.bnk	9/5/2021 7:14 PM	BNK File	6 KB
1.stm	9/5/2021 7:14 PM	STM File	1 KB
1.wsy	9/5/2021 7:14 PM	WSY File	5 KB
2.dat	9/5/2021 7:14 PM	DAT File	1 KB
3.dat	9/5/2021 7:14 PM	DAT File	1 KB
4.bdi	9/5/2021 7:14 PM	BDI File	1 KB
4.dat	9/5/2021 7:13 PM	DAT File	1 KB



JAMToolsDocumentation

.BNK	Instrument Bank
.WSY	WSYS
.STM	Stream Table
.BDI	Build Date Info
.DAT	Unknown data
.ARC	Sound or Sequence Archive
.BSTN	Binary Sound Table Namearchive
.BST	Binary Sound Table
.BMS	Binary Musical Sequence



JAMToolsDocumentation

Formats

AAF

The AAF format has two unique sections and a bit of a “cheat” to get it to function. The AAF format has a continuous table of sections, but has special markers to determine where to put the clusters for the wavesystems and instrument banks, as they are stored in a special way.

DO NOT PUT A WSYS OR IBNK DIRECTLY IN THE TABLE. It will butcher your AAF. (This only applies for the AAF format. The BAA format doesn't care.)

Here is what the “Cheat” looks like.



You can shift it wherever you like in the includes, so long as it appears. These will both put the contents of the “banks” includes list and the “waves” includes list in a special table format required by the game (respectively).



JAMToolsDocumentation

BAA

Different from the AAF type, BAA will not include a “waves” and “banks” cluster. Everything is listed as a raw include entry. The large difference is that the hashes are different.

```
{
  "originalFile": "GCKart.baa",
  "projectName": "Audio_Modding",
  "format": "BAA",
  "includes": [
    {
      "hash": 1651733536,
      "type": "SOUND_TABLE",
      "path": "include/0.bst",
      "uid": 0,
      "flags": 0
    },
    {
      "hash": 1651733614,
      "type": "SOUND_TABLE_STRINGS",
      "path": "include/1.bstn",
      "uid": 0,
      "flags": 0
    },
    {
      "hash": 2004033568,
      "type": "WSYS",
      "path": "include/2.wsy",
      "uid": 0,
      "flags": -1
    },
    {
      "hash": 2004033568,
      "type": "WSYS",
      "path": "include/3.wsy",
      "uid": 1,
      "flags": 0
    },
    {
      "hash": 2004033568,
      "type": "WSYS",
      "path": "include/4.wsy",
      "uid": 5,
      "flags": 0
    }
  ]
}
```



xayrga

JAMToolsDocumentation

Notes

Hashes

Hashes are raw data that get put into the AAF or BAA.

```
set(h) 00 01 02 03 04 05 06 07 08
00000 00 00 00 02 00 00 00 30 00
00010 00 00 00 00 00 00 00 03 00
00020 00 00 00 01 00 00 00 00 00
00030 49 42 4E 4B 00 00 08 E0 00
00040 00 00 00 00 00 00 00 00 00
```

This is a hash. You'll need to know the hash for each type that you want to add. Usually you can just copy them from other hashes in the project. For example, if you wanted to add a new IBNK in your AAF, its hash would have to be 2.

FOR AAF

IBNK	2
WSYS	3
SOUND_TABLE	1
SEQUENCE_COLLECTION	4
STREAM_MAP	5



xayrga

JAMToolsDocumentation

FOR BAA

SOUND_TABLE	1651733536
SOUND_TABLE_STRINGS	1651733614
WSYS	2004033568
IBNK	1651403552
MUSIC_SEQUENCE	1651340064



WSYSBUILDER



WSYSBUILDER



xayrga

JAMToolsDocumentation

About

WSYSBuilder, as it's name implies is used to pack and unpack WSYS's.

You can extract the .ws or .wsy files from your Audio Archive with JAMPACKED, or if you're using the complete toolset, you can find them in the "include" folder in the "Audio_Modding" folder.

WSYSBuilder was the first tool developed for JAMTools, so it's a bit different.

WSYSBuilder, unlike most of the tools here, supports the WSYS format for all games.

Command Line

This tool has built in help for it's commandline.

Please execute "wsysbuilder help"

```
Syntax:
wsysbuilder <operation> [args....]
wsysbuilder unpack    <wsFile>    <project file>
wsysbuilder pack      <projectFile> <wsOutput>

Optional arguments:
-encode-format <format> : Only when using 'pack', changes what format your custom sounds are saved into the .aw with
  Available Formats:
    adpcm4h1e -- Balance quality and compatibility, doesn't work on gamecube hardware.
    adpcm4    -- Has occasional clicking and some artifacts, but with a significant quality reduction. Compatible with hardware and LLE.
    pcm8     -- Best quality, compatible with real hardware, huge filesize, can be used only in moderation before sounds will stop playing.

-awpadding <bytes>      : Changes the padding inside of the .aw file (when repacking). Fixes compatibility with a few games.
-skip-transform         : Doesn't decode the .wav files whenever unpacking (must have an existing wavetable.json, used only for fast re-unpacking).
-awpath <path>          : Changes the directory to look for .AW files when unpacking.

https://www.xayr.ga/tools/
https://github.com/XAYRGA/wsytool/tree/wsbuilder
```



JAMToolsDocumentation

Project Output

The output of a wsysbuilder project looks like this.

DOLPHIN\root\mkdd\files\AudioRes\Audio_Modding\SelectVoice

Name	Date modified	Type	Size
custom	1/11/2022 8:03 PM	File folder	
ref	1/11/2022 8:03 PM	File folder	
scenes	1/11/2022 8:03 PM	File folder	
wav	1/11/2022 8:03 PM	File folder	
_README.txt	1/11/2022 8:03 PM	Text Source File	1 KB
manifest.json	1/11/2022 8:03 PM	JSON Source File	1 KB
wavetable.json	1/11/2022 8:03 PM	JSON Source File	9 KB

manifest.json

This is the “root” file for the project. It contains several fields to specify rebuilding parameters.

```
DOLPHIN > root > mkdd > files > AudioRes > Audio_Modding > SelectVoice > {} manifest.json > ...  
{  
  "id": 6,  
  "waveTable": "wavetable.json",  
  "sceneOrder": [  
    "scenes/SelectVoice_0.aw.json"  
  ]  
}
```



JAMToolsDocumentation

id

This will be the global ID of the WSYS when it is loaded ingame. Each WSYS that is loaded must have a unique ID.

wavetable

This is the file that will be used to load all of the waveid's for the entire WSYS. You cannot rebuild without this.

sceneOrder

This is an ordered list of references to "scene" files. The "scenes" generate the .AW files.

wavetable.json

The wavetable contains a list of waveid's and the associated information to load with the WSYS .

```
{  
  "40": {  
    "format": 0,  
    "key": 60,  
    "sampleRate": 32000.0,  
    "sampleCount": 46753,  
    "loop": false,  
    "loop_start": 0,  
    "loop_end": 46753,  
    "last": 32,  
    "penult": 0  
  }  
}
```



xayrga

JAMToolsDocumentation

The key of the dictionary is in fact the waveid it is associating that configuration information with.

```
1 },
2 "40": {
3     "format": 0,
4     "key": 60,
5     "sampleRate": 32000.0,
6     "sampleCount": 46753,
7     "loop": false,
8     "loop_start": 0,
9     "loop_end": 46753,
10    "last": 32,
11    "penult": 0
12 }
```

The ID lines up with the wav files in the “wav” folder.

MAKE SURE YOU READ THE NOTES.

format

This is the format that the sound is repacked in

0	ADPCM4
1	ADPCM2
2	PCM8
3	PCM16



JAMToolsDocumentation

key

This is the base midi note for the sound when it is played back. Raising it will lower the pitch, and lowering it will raise the pitch. It is 0-128. The note that plays gets subtracted by this value.

sampleRate

the samplerate of this sound. The rate at which it plays back. Technically this can also modify pitch, but it is not recommended.

sampleCount

The count of each individual sample for this sound until the end, regardless of loop point.

loop

Does the sound loop?

loopStart

What sample the sound starts to loop at.

loopEnd

What sample the looping jumps back to the loopStart at if loop is enabled.

last / penult

Part of the ADPCM4 rebuild system. It requires these two values to transition the ADPCM4 stream back into the loopStart.

A final note about all of these values.

If you are adding a custom sound from a .wav file, all values except “key” will be ignored and replaced with the information derived from the .WAV file at runtime, including loops. WAV files can be looped in most any DAW with a “SMPL” header.

Your JSON file will not be modified or saved over.



JAMToolsDocumentation

Scenes

Scenes are effectively the .AW files. The game will load a different .aw / scene depending on what level or scenario you're in currently. Each .aw file is custom built to contain the sound data that a level or scene will need, and only that data.

The basics of a scene are the "awfile" which controls the output name of the .aw, and of course the "waves". The waves are numbers, basically an array of "waveid"'s. Meaning any number in this list must have appeared inside of the "wavetable.json" first.

```
DLPHIN > root > mkdd > files > AudioRes > Audio_Modding > SelectVoice > scenes > (i) SelectVoice_0.aw.json > ...  
{"awfile": "SelectVoice_0.aw",  
 "waves": [  
  0,  
  1,  
  2,  
  3,  
  4,  
  5,  
  6,  
  7,  
  8,  
  9,  
  10,  
  11,  
  12,  
  13,  
  14,  
  15,  
  16,  
  17,  
  18,  
  19,  
  20,  
  21,  
  22,  
  23,  
  24,  
  25,  
  28,  
  29,  
  26,  
  27,  
  30,  
  31,  
  32,  
  33,  
  34,  
  35,  
  36,  
  37,  
  38,  
  39,  
  40  
 ]  
}
```



JAMToolsDocumentation

The folders and their function

WAV

The WAV folder contains the decoded version of all of the sound files, named by their ID. You can use this to listen to sounds and find the ID that you're looking for. This will not affect rebuilding if it is modified.

REF

The REF folder is used to fulfil a sound data request in the event that the file is not available in the "custom" folder. So if you don't replace a sound it will return to its normal self by using the information in this folder.

Custom

The custom .WAV files for sound replacement go in here. This is the first folder searched for a soundID during rebuilding. The .WAV files must share the same name as the soundID. If you put a file in here with the same name as an existing soundID, it will replace the audio contents of that ingame.

All WAV files must be 16 bit mono PCM when importing, 32khz or less. Importing will fail and stop if this is not the case.

Where do I find .ws / .wsys files?

You can look in the include folder of any extracted JAMPACKED project. The actual WSYS body is contained within the Audio Archive (BAA/AAF). So you'll have to extract it out with jampacked first, and then repack it after you've changed it with wsysbuilder.



BAST



BAST



xayrga

JAMToolsDocumentation

About

BAST is used to pack and unpack the sound table and sound table name files.

It is compatible with both JAudioV1 and JAudioV2.

Sound Table

The sound table is used to supply parameters and allocate slots for sounds either in the SE.BMS, Streamed files, or otherwise. This is where a sound is assigned its "ID". In order to ADD a sound you have to ADD a sound slot in the sound table.

Command Line

This tool has built in help for it's commandline.

Please execute "bast help"

```
bast unpack <bst file> <output folder>
bast pack <project folder> <bst file>
bast unpackbst <bst file> <bstn file> <output folder>
bast packbst <project folder> <bst file> <bstn file>

F:\TOOLSET\JTK\bast\bast\bin\Release>_
```

Please take notice of the second format. BAST supports both JAudiov1 JASE format and JAudiov2 BST format. You need to use the appropriate format for the game you are modifying.

Sound ID's

Sound ID's are assigned to a category and sequentially. Each sound a has a "Global" id, but the times that is actually used is nil.



JAMToolsDocumentation

Categories are loaded in a particular order with a particular set of sound ID's that it loads.

The structure of a sound ID is a u16 (c= category, f = flag, n = id)

c(f+n)nn








So sound 61 in category 11 would look like B83D (flag for not empty is 800, if the sound is empty it would be B03D, but we add 800 because it is an allocated sound. Any DUMMY sounds will have 0 instead of 8)

When repacking, BAST will automatically calculate sound ID's.

Format here <https://xayr.ga/wiki/BST>

JASE AND THE CATEGORY INCLUDES

The JASE or V1 format is composed of just categories and sounds.

PC > Doc (F:) > TOOLSET > JTK > bast > bast > bin > Debug > test > includes				
Name	Date modified	Type	Size	
 0x1.json	12/19/2020 11:37 PM	JSON File	26 KB	
 0x2.json	12/19/2020 11:37 PM	JSON File	60 KB	
 0x3.json	12/19/2020 11:37 PM	JSON File	97 KB	
 0x4.json	12/19/2020 11:37 PM	JSON File	11 KB	
 0x5.json	12/19/2020 11:37 PM	JSON File	146 KB	
 0x11.json	12/19/2020 11:37 PM	JSON File	24 KB	
 0x12.json	12/19/2020 11:37 PM	JSON File	1 KB	

BAST will generate .json files with the category ID on each one. Inside of the JSON files will be sound configuration.



JAMToolsDocumentation

Sound ID's and Category Data

Each of the entries consumes a sound ID. You can have a maximum of 2047 sounds per category.

```
{
  "id": 1354,
  "index": 543,
  "name": null,
  "sflags": 0,
  "pflags": 0,
  "uflags1": 0,
  "uflags2": 0,
  "type": 128,
  "loadMode": 0,
  "is_not_empty": false,
  "pitch": 1.0,
  "volume": 65280
}
```

You can copy and paste this to make new sound slots, the parameters when adjusted will affect that particular sound ingame.

ID is the global ID for that sound (remember that it will usually be in hexadecimal)

Index is the ID inside of that category, remember $((\text{categoryID}) \ll 12) + 800 + \text{soundID}$ will be the ID of that sound. "ID" and "index" are just helper values.

Sflags, and uflags are unknown, only there for rebuilding

Type indicates the type of sound, different values here indicate how the sound is played back



JAMToolsDocumentation

Loadmode indicates the source for the sound

Is_not_empty indicates whether or not the sound should be treated as an empty or not,

Pitch and volume are pretty self-explanatory.

REBUILDING JASE

When rebuilding a JASE-format archive, you'll need the rebuild hash or rebuild instructions. This is a giant hexadecimal number that indicates the order of categories..

```
bast.exe pack my_bst_project out.bst "FFFFFFFFFFFFFFFF1011000102030405060708"
```

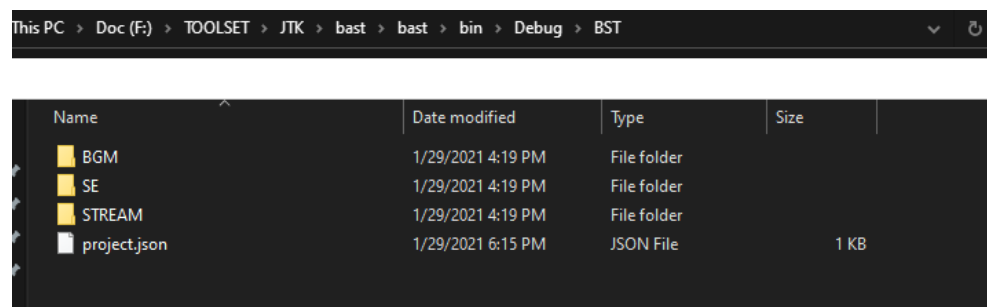
This will pack category 10, then 11, then 0, 1,2,3,4,5,6,7,8. Every game requires that its category order is intact. If you can't create this, you can always pass the parameter as "guess" or it will default to it.

BST format

BST format is usually seen in "v2" jaudio implementations.

Contrary to JASE, the BST has "libraries" and "Categories".

In this sense, "Categories" refer to a type of sound. Libraries refer to a list of sounds with a common function.



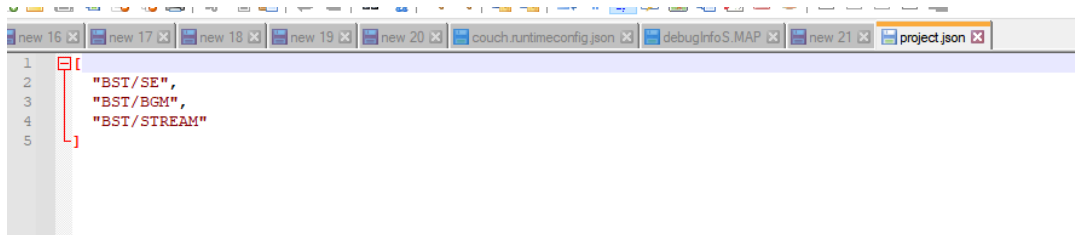
Name	Date modified	Type	Size
BGM	1/29/2021 4:19 PM	File folder	
SE	1/29/2021 4:19 PM	File folder	
STREAM	1/29/2021 4:19 PM	File folder	
project.json	1/29/2021 6:15 PM	JSON File	1 KB



JAMToolsDocumentation

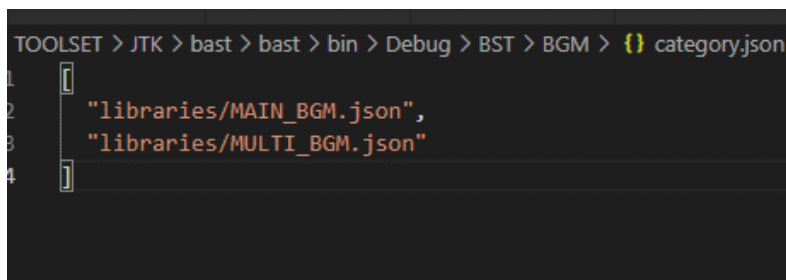
Project.json

The project.json will list a list of folders that contains the BST data, this is a list of “categories”



```
1 [
2   "BST/SE",
3   "BST/BGM",
4   "BST/STREAM"
5 ]
```

Inside each of the listed folders will be a “category.json” that will list the libraries that were included in this category.



```
TOOLSET > JTK > bast > bast > bin > Debug > BST > BGM > {} category.json
1 [
2   "libraries/MAIN_BGM.json",
3   "libraries/MULTI_BGM.json"
4 ]
```

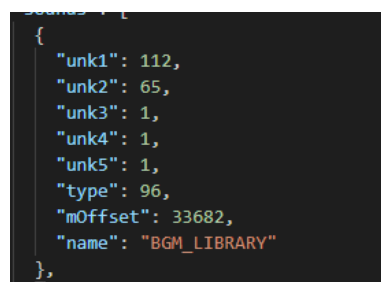
Library JSON files & Sounds

The library JSON files will contain the parameters for sound playback.

The BST format was updated to have unions for sound parameters, so you’ll see some common parameters through sound objects, but you’ll also find that the various types have different parameters. Note the differences between a streamed and sequenced sound entry.



```
{
  "unk1": 128,
  "unk2": 90,
  "streamType": 14,
  "streamPath": "/AudioRes/Stream/SMG_title_strm.ast",
  "type": 112,
  "mOffset": 35028,
  "name": "STM_TITLE"
},
```



```
{
  "unk1": 112,
  "unk2": 65,
  "unk3": 1,
  "unk4": 1,
  "unk5": 1,
  "type": 96,
  "mOffset": 33682,
  "name": "BGM_LIBRARY"
},
```



xayrga

JAMToolsDocumentation

Not all values have been figured out for these parameters yet, so they will change as the tool evolves. Usually, you can get the desired results just by editing one of the named variables.

Sound ID's

Sound ID's in V2 (BST) are calculated a bit differently)

Again, as the last engine has, sound ID's are in fact sequential in the category. Meaning the next sound will consume the next ID.

However, in this version of the engine, you can have 65534 sounds per category.

Sounds have both a global ID and a local ID, but only the local ID is used (based on category).

The local ID of a sound can be calculated with the following code

```
SoundIndex | (LibraryIndex << 0x10)
```

When repacking, BAST will automatically calculate sound ID's.

