COD Sound Engine: Mod Tools

Adding New Sounds to a Map (Zone)

To add new sounds to a map start by editing the sound zone configuration (.SZC) file that gets generated when the map is created in the mod tools launcher.

```
<game>\usermaps\mp_mynewmap\sound\zoneconfig
```

The SZC file is basically just a way of mapping sounds, defined in various .CSV files, to the higher level .ZONE files that the engine uses to specify which assets are included in a fast file. The SZC file is included in the .ZONE file for the map as a SOUND entry.

```
 sound,mp_mynewmap
```

There is an example sound alias CSV file that you can use to get started adding sound aliases. What a sound alias is will be described next but for now think of it as a “sound” you are adding.

```
<game>\share\raw\sound\aliases\user_aliases.csv
```

All user created sound aliases can be put in this example CSV and added as a sound source to the map’s SZC file in the sources array.

The important fields in this file that need to be filled out correctly are:

Sources – this is an array of “Source” types.

Source

> **Type** – set to ALIAS for adding an entry pointing to a sound alias CSV.
>
> **Name** – just the name of the new source. Should match the base of the filename specified in the Filename field. Example, given a Filename “`mp\\mpc_my_mp_map.csv`”, the Name field should be specified as “`mpc_my_mp_map`”
>
> **Filename** – the path to the .CSV file that contains the sound aliases, relative to `<game>\share\raw\sound\aliases\`

```
"Sources" : [
{
 "Type" : "ALIAS",
 "Name" : "user_aliases",
 "Filename" : "user_aliases.csv",
 "Specs" : [ ]
}
]
```

Be careful with the syntax of this file and make sure that the name matches the base filename of the actual CSV file minus the CSV extension.

If multiple maps are created and/or a lot of different types of customs sounds are added then it will make sense to create a copy of this example CSV to break things up logically. Each CSV that is to be loaded into the map will need to be a separate entry under the sources array. Be careful with syntax and add a comma after each source CSV.

```
"Sources" : [
{
 "Type" : "ALIAS",
 "Name" : "user_aliases",
 "Filename" : "user_aliases.csv",
 "Specs" : [ ]
},
{
 "Type" : "ALIAS",
 "Name" : "user_aliases1",
 "Filename" : "user_aliases1.csv",
 "Specs" : [ ]
},
```

The CSV files pointed to by the sound sources in this file are basically a collection of “sound aliases”. Sound alias is the term used when discussing a sound within the COD engine. It is essentially the set of data that completely describes a sound within the game including the path to a physical WAV file (Filespec) as well as numerous fields of metadata which control playback parameters of the sound at runtime. There are many parameters which can be used to control and manipulate each sound. The important ones are listed below.

As long as everything is filled out correctly the sounds listed in the CSV files that are referenced as sources will be available in the new map. Careful attention to maintaining the syntax of the SZC files is important.

Sound Aliases

As mentioned earlier, a sound alias is the term used to describe a sound in the COD BO3 engine. Sound aliases are added to the sound alias CSVs and the alias name (string) used throughout the engine to trigger sounds. Sounds are never directly referenced by their WAV filename. The whole idea of an alias is to associate a "sound" with a descriptive text name which can easily be referred to, passed into code/script functions, triggered from animations, script, within triggers in the map etc without any dependence on a particular physical WAV file.

Referring to sound aliases in this way provides an abstraction between the code/script/game data triggering the sound and the sound data itself, allowing the WAV file or playback parameters to be swapped out or tweaked by the sound designer without changing the triggers or references in the game. In addition, a one-to-one mapping of sound alias to WAV file is not always guaranteed (or desired). If one is to create aliases with identical names then the sound engine will treat this as a random situation and will choose one alias randomly from the potential pool of same named aliases. Similarly, one can enter a folder name rather than a single WAV filename in the FileSpec column. In this case the engine will choose one WAV file randomly from the folder specified allowing for variation (put 4 different bird chirps in a folder and have it pick a different one each time the alias is triggered).

Templates

In user_aliases.csv there is an example sound alias called "test_sound". While looking at this example alias notice that only a few of the fields are actually filled in with data out of the 78 possible columns. We have only specified the name, filespec (path to the physical wav file on disk relative to `<game>\sound_assets`), template, VolMin, and VolMax. In this case we are leveraging the power of a "template alias" which allows an alias to be created that contains certain properties that will be included by any custom aliases. In this example alias we are referencing a template "UIN_MOD". In `<game>\share\raw\sound\templates\template_mod.csv` there is a single template alias specified (UIN_MOD). The necessary parameters for creating an alias are specified here and are applied to each sound alias you create when you include this template. Values can be overridden from the template on a per-alias basis by just entering a new value for that column in the actual alias CSV.

Secondaries

A single call to trigger a sound alias can result in the playback of 0 to 10 sounds depending on how the aliases are set up and which banks are currently loaded. To "chain" aliases together and have one alias call another one simply add an alias name in the "Secondary" column of the original alias. When the original alias is called it will additionally attempt to playback the alias listed in the secondary column. This is particularly useful for sound events that trigger complex layers of sounds like weapon fire.

Alias Fields

All of the possible columns that can be tweaked in the CSV files are listed below. Many of these columns are for more advanced users and will likely never need to be touched. It is strongly recommended to create one to a few templates to use with some parameters that work and then include those templates in any user created aliases, tweaking things like volumes and other simple parameters as needed. The default value for each column is in parentheses after each column name. If this field is not specified in either the template or the alias this is the value that will be used.

Name (“”) – name of this sound alias. A text string to be used throughout the engine to trigger sounds. Multiple aliases of the same name that are built into a single sound zone will be treated as a random situation when they are triggered.

Storage (loaded) – specifies whether the sound is loaded (specify “loaded”) into RAM or streamed (specify “streamed”), from disk. Loaded sounds will go in the map’s *.sabl banks and streamed sounds will go in the *.sabs banks.

FileSpec (“”) – the path and filename of the physical wav file for this sound alias relative to `<game>\share\raw\sound\aliases\.` Specify a folder name to have the sound engine randomize between all wav files in that folder when this alias is triggered.

FileSpecSustain (“”) - If this column is filled out with a looping asset then it will be triggered when the (one shot) asset specified in FileSpec finishes.

FileSpecRelease (“”) – If this column is filled out then this asset will be triggered when the looping asset specified in FileSpecSustain is stopped.

Template (“”) – As explained above, this field points to the name of a template alias defined in a template CSV located in `<game>\share\raw\sound\templates\.`

Loadspec (“”) – A name of a loadspec contained in `<game>\share\raw\sound\globals\loadspec.csv.` This is simply a way of assigning a grouping of certain sounds to be included into a sound zone by specifying the loadspec along with the CSV filename in the SZC file. To bring in all of the aliases in a CSV into an SZC it is not necessary to specify any loadspecs. To bring in 3 aliases that have been tagged with the loadspec “surf_gravel” make sure this loadspec name is specified for the loadspec field for the aliases and include the alias CSV in the SZC file and also specify this loadspec name under “Specs” . In this case it will only bring in those 3 aliases.

Secondary (none) – specify another sound alias here and it will be triggered immediately after the “primary” sound alias. You can chain together multiple secondaries to achieve multiple sounds per sound alias trigger. Weapons are handled in this manner.

Bus (bus_fx) – This is the bus that the sound belongs to. Some of these busses are used to control the volumes on groups of sounds. For example, in the sound menu options sliders.

BUS_FX
BUS_VOICE
BUS_PFUTZ
BUS_HDRFX
BUS_UI
BUS_MUSIC
BUS_MOVIE
BUS_REFERENCE

VolumeGroup (none) – specify one of the values from `<game>\share\raw\sound\globals\volume_group.csv.` This basically allows you to group together like sounds and specify an attenuation scaler to that group. You can adjust these per game mode in the volume_group.csv.

Duck (“”) – Specify a duck name that will be triggered when this sound alias plays back. A sound duck will lower (duck) the volume of various groups of sounds in the engine. Each sound alias specifies a duck group that it belongs to. See the list of possible ducks in `<game>\share\raw\sound\ducks.`

DuckGroup (“”) – The name of the duck group that this sound alias belongs to (as discussed above). The list of duck groups that a sound can specify is located in `<game>\share\raw\sound\globals\duck_group.csv.`

ReverbSend (0) – value in dB SPL that the reverberated (wet) portion of the sound will be attenuated by. 100 = all of the signal is mixed into the reverb signal path, 0 = none of the signal is mixed into the reverb signal path.

CenterSend (0) – override the center value specified in the pan for this alias if the specified CenterSend is larger (dB SPL).

VolMin/**VolMax** (92 db SPL) – specify the volume min and max of the sound in dB SPL and the engine will randomize between these levels.

DistMin (0)/**DistMaxDry** (10000)/**DistMaxWet** (10000) – for a 3D sound, the DistMin is the point in distance units where the sound will begin to "falloff" or attenuate. So the sound will be at max volume from the player until it reaches this distance. DistMaxDry is the point where it will become silent. It will scale between full volume at DistMin to silent at DistmaxDry. DistMaxWet is provided to give similar behavior but for the "wet" or reverberated portion of the sound. This allows you to extend the distance in which you will hear the sound reverberating.

DryMinCurve (allon)/**DryMaxCurve** (default) – this is the curve type that specifies how the sound will falloff depending on distance. Sound does not attenuate linearly proportional to distance so for many sounds you may want to specify a custom curve to control this. Using the defautls will probably be fine 99% of the time. The DryMinCurve applies to the attenuation from listener (player) to DistMin and the DryMaxCurve applies to the attenuation between DistMin and DistMax.

WetMinCurve (allon)/**WetMaxCurve** (default) – same as the above dry signal but for the reverberated portion of the sound.

LimitCount (8) – the number of simultaneous sounds of this alias that will play back before limiting will occur.

LimitType (oldest) – Once the engine hits the LimitCount for this particular alias a decision needs to be made depending on this type specified.

none	No limiting. Count is ignored and the sound plays back.
oldest	Oldest sound of this alias will be stopped immediately.
reject	This sound alias will be rejected. All others continue.
priority	The sound instance to be stopped is chosen depending on priority.

EntityLimitCount (8) – Same as LimitCount except rather than taking all playing sounds into account it will limit the instances of this sound alias playing back on a particular entity.

EntityLimitType (oldest) – Same as LimitType except specifies the limit type to be applied to a particular entity.

PitchMin/**PitchMax** (0) – specify a min/max pitch in cents for this sound instance to play back at. It will randomize between min/max pitch. +1200 = double pitch, -1200 = half pitch.

PanType (2d) – Specifies how the sound changes depending on world position

2d	Constant volume/pan. Plays full volume and ignores distance. Music, UI sounds etc.
3d	Changes volume/pan depending on world position.
2.5	Ignores distance but pans sound around the player depending on world position.

Pan (default) – specifies the name of a pan type in `<game>\share\raw\sound\globals\pan.csv` which determines how the sound will be panned throughout the 3D sound field. Refer to this CSV for the possible values and how the sound will be played back in each speaker.

Futz – The name of a futz patch that will be applied to this sound in real-time. `<game>\share\raw\sound\globals\futz.csv` contains a list of possible futz patches that can be included.

Looping (nonlooping) – specifies whether the sound plays once (nonlooping) or requeues at the end and continues playing until it is stopped (looping).

RandomizeType (“”) – If a value is specified here the engine will compute the same random value each time this sound is called on the same entity. If you want actual randomness per entity this field should be left blank. Useful for example if you want to randomize between PitchMin/PitchMax for weapons to give some slight difference to each AI shooting at you but want to maintain similar sounds for each individual entity (AI character). Leaving it blank will simply randomize for each instance of the sound.

volume	Same random volume if sound called on same entity
pitch	Same random pitch if sound called on same entity
variant	Same random alias variant selection of called on same entity

Probability (1.0) – a specified percentage of the sound playing back or not after it’s triggered. 1.0 = 100%, 0.0 – 0% (no playback).

StartDelay (0) – value in milliseconds to delay the start of the sound once it is triggered

EnvelopMin (0)– When the listener (player) is within this distance from a 3D sound source the sound will be panned into the speakers that would not normally have any of the signal. The amount of signal that is panned into these speakers is determined by the EnvelopPercent. Example: if a sound is directly in front of the player it would typically be panned equally between the left and right speaker.If an EnvelopMin is specified and the player is within that radius the amount of EnvelopPercent (in dB SPL) is panned into the rear and side channels.

EnvelopMax (0) – This distance defines an outer ring around the sound source where the amount of EnvelopPercent is then scaled linearly between the full value and 0 as the listener moves from the min to the max ring.

EnvelopPercent (0)– As mentioned above, the amount of the signal (in dB SPL) that is panned into the channels to produce the envelop effect.

OcclusionLevel (0.25) – float value between 0-1.0 that specifies how much the real-time occlusion level will be scaled. 1.0 would mean 100% of the calculated occlusion value would be applied, 0.5 means the calculated occlusion value would be scaled 50%.

IsBig (no) – If set to true this turns off geometry based occlusion when you are close to a 3D sound.

DistanceLpf (yes) – whether distance based low passed filtering will be applied to this sound. Engine will take the lesser of the calculated distance based LPF value and the occlusion based LPF value.

FluxType (none) – Flux is a system in the sound engine that allows a sound designer to get 3D positional effects on a sound without having it tied to an actual 3D object in the world. A good example where we use this is on grenades. The explosion sound will play at the location where the grenade exploded but we can add “flux” to the sound to tell it to travel away from the original location it triggered. For the grenades we can have the “debris” layer move away from the explosion as it plays to simulate the effect of debris falling away from the explosion over time. The table below shows the various flux types which determine the direction the sound will “travel” due to flux.

none	No flux
left_player	To the left of the player
right_player	To the right of the player
center_player	Straight out from the center of the player
random_player	Random direction from the player
left_shot	To the left of a shot
center_shot	Straight out from the center of the shooter
right_shot	To the right of a shot
random_direction	Completely random in 3D space

FluxTime (0) – time in milliseconds that the flux effect will be applied for as it travels to DistMaxWet.

Subtitle (“”) – any subtitle that should be displayed on screen, usually accompanying a dialog line.

Doppler (no) – yes/no whether this sound will have the Doppler effect applied.

ContextType/ContextValue – Do not use these.

Timescale (no) – yes/no whether the sound should be effected by the global timescale setting.

IsMusic (no) – yes/no depending if the alias is part of the music system.

IsCinematic (no) – yes/no depending if the alias is part of a cinematic sequence.

FadeIn (0) - time in milliseconds that the sound will fade in from silence to full volume.

FadeOut (0) – time in milliseconds that the sound will fade out over when it ends. Fades out from currently playing volume to silence over this time.

Pauseable (yes) – yes/no depending on if the sounds should pause when the game pauses. Should be “yes” for almost everything except for sounds you want during the pause menu.

StopOnEntDeath (no) – yes/no to determine if this sound should respond to a call to stop sounds on an entity.

Compression (100) – consoles only as PC is FLAC encoded at a constant compression ratio. This field allows specification between 0-100 of a compression “quality” where 100 is least compressed and 0 is most compressed.

StopOnPlay (“”) – not currently implemented.

DopplerScale (1.0) – value between -100.0f – 100.0f that scales the Doppler effect on a sound.

FutzPatch (“”) – name of a futz patch defined in `<game>\share\raw\sound\globals\futz.csv` to apply to the sound.

VoiceLimit (no) – unused

IgnoreMaxDist (no) – By default the engine will cull a sound play event if a 3D sound is triggered on an object that is already past the maximum distance specified (DistMaxWet). Set to true to ignore the distance culling check. Useful for one shots that start far away from but approach the player. Example a plane by.

NeverPlayTwice (no) – Set to yes to prevent a sound alias from triggering more than once. Useful to limit repetitive voice over lines that trigger multiple times. The play flag resets on a new level or restart from checkpoint (death).

ContinuousPan (yes) – used to stop any non-looping, 3D sounds when the listener snaps to a different position over a specified threshold (512 distance units) in a single frame. For example the player is warped to a different position in the map. This prevents sounds from popping when they are naturally stopped due to distance. Set to no to prevent a sound from stopping in this case.