Vector Breach

Audio Implementation Master Document

Unreal Engine 5 FPS Audio Redesign | MetaSounds | Blueprints | Dynamic Mixing

Field	Details
Author	Conor Dawson
Project type	Game audio implementation and technical audio portfolio project
Engine / tools	Unreal Engine 5.7, MetaSounds, Blueprints, Sound Classes, Sound Mixes, Submixes, Audio Insights, Ableton Live
Draft version	V1.1
Date	2026-04-26

Contents

1. Executive summary

Vector Breach is a focused Unreal Engine 5 FPS audio implementation project. The work demonstrates how a full audio pass can be planned, built, debugged, mixed, and validated inside Unreal using native tools rather than middleware.

The main value of the project is not isolated sound replacement. It is the complete technical-audio layer: MetaSound systems, Blueprint audio hooks, environmental emitters, local-versus-world playback decisions, dynamic SoundMix behaviour, clean SoundClass/Submix routing, audio profiling, and production-style asset maintenance.

1.1 What the project demonstrates

- MetaSound-based systems for weapons, footsteps, impacts, UI, ambience, environmental props, and gameplay feedback.
- Blueprint integration into an existing FPS framework without disrupting gameplay logic.
- Local player feedback separated from TPP/world presentation for gunshots, movement, restock, low-health, and fall-pain style events.
- Environmental ambience driven by state, height, cover/interior triggers, proximity emitters, and dedicated audio occlusion.
- SoundClass/Submix architecture that exposes readable stems in Audio Insights and supports category-level mix control.
- Dynamic mix control using SoundMix assets and selective passive Sound Class triggers.
- Practical validation using in-game listening, Audio Insights, isolated MetaSound renders, offline WAV analysis, and final gameplay capture review.
- Production discipline through audit scripts, reference repair, unused-asset cleanup, SoundWave naming/folder review, and implementation documentation.

Vector Breach is an audio implementation portfolio project built using a licensed Unreal Engine FPS template. The authored work documented here is the audio redesign and implementation layer: MetaSounds, Blueprint audio hooks, routing, dynamic mixing, validation, and final presentation.

2. Implementation overview

The project is built around repeatable audio patterns: MetaSounds generate and vary sound content; Blueprints decide when and how sounds are triggered; Sound Classes provide category-level control; Submixes expose stems and final processing; SoundMix assets handle dynamic state and event relationships.

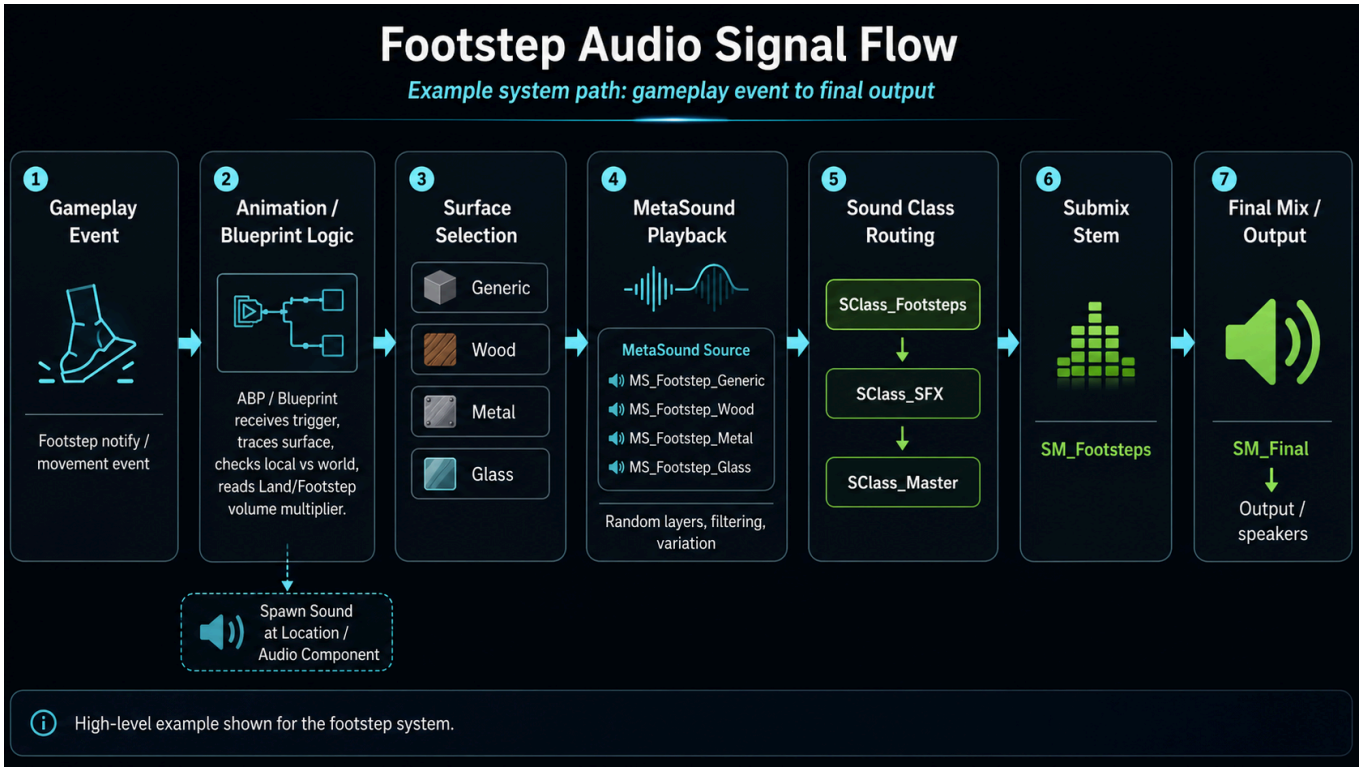
2.1 Signal-flow model

Layer	Primary assets / tools	Role
Source design	Ableton Live, exported WAV layers	Create, edit, layer, transient-align, and render the raw sound material before import.
Runtime sound generation	MS_* MetaSounds and MSP_* patches	Randomisation, layer selection, surface selection, suppressed/unsuppressed branches, tail selection, filtering, and parameter-driven variation.
Gameplay integration	BP_PlayerCharacter, BP_BaseWeapon, ABP_Base / ABP_Base_TPP, BP_AmbienceManager, emitter Blueprints	Trigger audio, set parameters, manage local-only events, handle overlap/state logic, and connect audio to existing gameplay events.
Routing and mix control	SClass_*, SM_*, MIX_* assets	Stem routing, Audio Insights metering, dynamic mix states, low-health processing, UI priority, environment ducking, and final output control.
Validation and maintenance	Audio Insights, Unreal Insights, audit scripts, WAV analyzer, render/capture review	Verify runtime behaviour, isolate loudness issues, repair references, remove unused assets, and support final documentation.

2.2 Design principles used throughout the project

- Balance individual MetaSounds first, then use Sound Classes and SoundMixes for category-level or state-level correction.
- Use local-only playback for direct player confirmations where spatialization would make the feedback less clear.
- Keep FPP/local and TPP/world presentation separate when player perspective changes the job of the sound.
- Avoid broad mix triggers. Use child Sound Classes for only the events that should drive dynamic mixing.
- Do not push every quiet sound louder. Solve readability with source shape, spectrum, routing, attenuation, and background-space control.

- Prefer lean, shippable implementation over fragile architecture when the simpler version solves the game-audio problem.



3. Environmental audio systems

The environmental audio pass created reusable Blueprint and MetaSound systems for ambience state, perimeter traffic, proximity loops, interactive foliage, and clean audio occlusion. The main objective was to make the map feel authored rather than filled with disconnected loops.

3.1 Ambience manager and ambience-state patch

BP_AmbienceManager owns the high-level ambience state. Interior and under-cover trigger volumes feed target values, smoothing prevents abrupt changes, and MSP_AmbienceState centralizes the math used for indoor contribution, wind reduction, and cutoff movement. MS_Amb_Bed_Outdoor uses that patch so the core state behaviour is not duplicated inside every ambience asset.

> **Backup:** *See Appendix C for the full BP_AmbienceManager EventGraph, RegisterVolume function, and MSP_AmbienceState MetaSoundPatch. The appendix includes full graph overviews.*

3.2 Outdoor bed, height, cover, and traffic

MS_Amb_Bed_Outdoor blends the main outdoor bed, rooftop/big-wind layer, and an indoor contribution layer. HeightLevel and WindIntensity alter the outdoor balance. InteriorState and UnderCoverState reduce wind and pull filtering down when the player moves under cover or inside.

MS_Traffic_Perimeter and BP_TrafficEmitter provide low-level perimeter traffic around the edge of the map. Traffic is updated separately from the main bed so it can react to indoor/under-cover states while still remaining spatially placed.

> **Backup:** *See Appendix D for the full MS_Amb_Bed_Outdoor graph, traffic emitter placement examples, traffic attenuation settings, and Audio Insights evidence showing SM_Ambient activity during runtime playback.*

3.3 Proximity emitter families

Emitter family	Blueprint pattern	MetaSound/content	Implementation notes
Bush rustle	BP_BushEmitter with movement-aware overlap logic	MS_Bush random one-shots	Uses speed, accumulated distance, min-step distance, cooldown, play-state checks, and concurrency to avoid spam and cutoffs.
Computer desk electronics	Sphere overlap with InsideCount; FadeIn on 0->1, FadeOut on 1->0	MS_ComputerDesk loop	Very close attenuation. Multiple desks can layer without becoming one global loop.
Light hum	BP_LightHumEmitter with MeshOverride and proximity fade logic	MS_Lights	Designed for many lights. Uses AudioOcclusion where wall muffling is useful.
Small fan	BP_FanEmitter overlap-gated loop	MS_Fan	Random start offset was removed during debugging after loop behaviour proved inconsistent.
Large fan	BP_LargeFanEmitter overlap-gated loop	MS_LargeFan	Used for larger rooftop/industrial fan presence and range validation.
Electric box / flag / prop sweeteners	Reusable loop or proximity-placed actor logic	MS_ElectricBox, MS_Flag, related prop loops	Adds localized identity without making the full map ambience carry every detail.

3.4 AudioOcclusion and collision cleanup

A dedicated AudioOcclusion trace channel separates audio muffling from gameplay Visibility traces. Structural geometry blocks AudioOcclusion; non-occluding props and audio trigger volumes ignore channels that would interfere with weapon traces, camera behaviour, or other gameplay systems.

This became a project-wide cleanup pattern: audio trigger colliders should be Query Only, overlap Pawn, and ignore gameplay traces unless there is a specific reason not to.

> **Backup:** *See Appendix E for the AudioOcclusion collision-channel setup, wall mesh response examples, attenuation occlusion trace-channel settings, and representative emitter collision settings showing that audio trigger logic is separated from gameplay traces.*

4. Weapon and combat audio

Weapon audio was focused on a polished assault rifle and pistol set. This kept the authored combat identity coherent and avoided spreading work thinly across unfinished weapons.

4.1 Weapon coverage and loadout control

The weapon roster was narrowed through prototype gating and loadout handling. Unsupported weapons were hidden, fallback-routed, or avoided so the authored audio work stayed aligned with the playable presentation. AC_Loadout was corrected so the front-end did not automatically populate multiple unwanted default loadout slots.

4.2 Assault rifle implementation

The rifle shot was built as a layered MetaSound rather than a single baked one-shot. The design separates low body, punch, mechanical layer, suppressed content, exterior tail, interior tail, and tail sweeteners. Parameters such as UsesSilencer and IsInterior switch content paths without duplicating the entire graph.

The rifle FPP unsuppressed shot was used as the main combat loudness anchor. TPP/world presentation was treated separately so remote/NPC weapon fire remains readable but does not dominate the player mix.

> **Backup:** *See Appendix F for the full MS_WPN_AR_Shot_FPP overview, close-up callouts for layer arrays, suppressed logic, IsInterior tail selection, output mixing, and supporting MS_WPN_AR_Shot_TPP attenuation / concurrency evidence.*

4.3 Pistol implementation

The pistol follows the rifle architecture but is lighter, shorter, and less low-mid heavy. Unsuppressed and suppressed variants use the same overall parameter-driven approach, while TPP presentation remains tighter and less dominant than rifle world fire.

4.4 Local versus world weapon presentation

Category	Example assets/classes	Treatment	Reason
Local FPP gunshots	SClass_Gunshots_Local; rifle/pistol FPP MetaSounds	Clear, present, drives local combat-focus ducking.	Prioritizes player weapon feedback.
TPP/world gunshots	Rifle/pistol TPP MetaSounds	Distance-aware attenuation, air absorption, occlusion, no broad local duck trigger.	Prevents enemy or NPC fire from making the whole map pump.
Reloads and handling	ClipIn/ClipOut, zoom, body movement	Short tactile one-shots below weapon-shot priority.	Adds mechanical detail without pretending to be a combat transient.
Bullet cases	Rifle and pistol case MetaSounds	Tight attenuation and concurrency.	Secondary detail near the weapon, not map-wide information.

4.5 Reload interruption and weapon-change fix

A reload interruption issue was solved by adding StopWeaponAnims on the CurrentWeapon reference before M ChangeWeapon in the player weapon-change paths. This prevents late ClipIn notifies from firing after the player cancels a reload by swapping weapons.

> **Backup:** *See Appendix G for the BP_PlayerCharacter change-weapon graph showing StopWeaponAnims before M ChangeWeapon, plus reload animation notify timing evidence for ClipOut and ClipIn.*

5. Movement, Foley, footsteps, and landings

Movement audio was rebuilt around surface material routing, local/world presentation, body Foley detail, and a separate landing system. The key decision was to keep each movement job clear rather than folding everything into one overloaded footstep asset.

5.1 Surface-based footsteps

Footsteps resolve through physical material selection into material-specific MetaSounds for Generic, Wood, Metal, and Glass. The system was debugged around an earlier issue where indoor tonal changes came from attenuation occlusion rather than intended footstep design. Once that was identified, occlusion and trigger setup were separated from gameplay traces and tuned more deliberately.

Local and world footsteps share content where useful, but attenuation, concurrency, and presentation do not need to be identical. This keeps the player movement readable without making every remote step equally prominent.

> **Backup:** *See Appendix H for ABP_Base / ABP_Base_TPP surface-material routing, a representative MS_Footstep_Generic array graph, local/world attenuation and concurrency assets, reusable random-player patch settings, and runtime capture showing footsteps across different surfaces.*

5.2 Randomisation and anti-repeat behaviour

The footstep arrays use a reusable random-player approach. The accepted practical configuration improved obvious repetition without becoming a full shuffle-bag implementation. This was a deliberate production tradeoff: acceptable variation with less risk and less complexity.

5.3 Landing system

Landing was kept separate from footsteps to avoid destabilizing the working footstep setup. Event OnLanded stores LastLandSpeedAbs, maps landing speed into LandVolumeMultiplier, preserves the existing damaging-fall logic, and routes landing sound selection through a trace below the player.

Four material landing MetaSounds are used: MS_Land_Generic, MS_Land_Wood, MS_Land_Metal, and MS_Land_Glass. Local and world presentation can be split through separate attenuation/concurrency settings.

Landing element	Current behaviour
Speed capture	LastLandSpeedAbs = absolute Velocity Z at landing.
Volume multiplier	Maps roughly from normal jump to heavier safe fall, with damaging falls handled separately by gameplay logic.

Trace	Trace below the pawn to select surface material and matching MetaSound.
Presentation split	Local landing is immediate; world landing can use attenuation and moderate occlusion.

5.4 Body Foley and movement detail

Body Foley, hand swipes, jump movement detail, and weapon handling sit below weapon and UI priority. These elements support embodiment and motion without being pushed to unrealistic solo-audition levels. A previous quiet/filtered body-Foley issue was traced to attenuation occlusion rather than the MetaSound source itself.

6. UI, objective, restock, health, and state feedback

UI and state feedback were organized by gameplay importance. Routine UI should not clear the whole mix. Combat/objective confirmations and player-state transitions need stronger priority, but only when they genuinely carry information.

6.1 UI hierarchy

Hitmarkers, headshot markers, objective notifications, countdowns, menu highlights, and kill/event confirmations were balanced as separate information tiers. The late-pass mix reduced the strongest UI assets after isolated analysis showed some elements were too hot relative to weapons.

UI group	Mix role	Guidance
Menu highlights	Routine navigation feedback	Keep subtle and below combat UI.
Regular hitmarker	Combat confirmation	Readable but below headshot marker and weapon transient.
Headshot hitmarker	Higher-value combat confirmation	Stronger than regular hitmarker, but not a weapon replacement.
Objective/countdown/result UI	Priority game-state information	Can trigger UI-priority ducking or use a priority child Sound Class.
Music stingers	State transition support	Should clear space selectively, not dominate normal gameplay.

6.2 Low-health state system

Low health is treated as a state, not just a damage event. BP_PlayerCharacter handles On Health Changed, gates to the actual local player pawn, fires a short one-shot only when crossing into low health, enables a persistent SoundMix while critical, and disables it after recovery hysteresis.

Element	Behaviour	Reason
Entry threshold	OldHealth > 60 and NewHealth <= 60	Triggers the one-shot only when entering low health.
Exit threshold	NewHealth >= 65	Prevents on/off chatter near the threshold.
Local-player gate	GetPlayerPawn(0) == Self	Avoids NPC/listen-server false positives.
Persistent state	MIX_LowHealth push/pop	Shapes the mix while the player remains critical.
Death handling	Clear low-health state immediately	Death cue/state takes priority.

Backup: *See Appendix I for the BP_PlayerCharacter On Health Changed sequence, local client events for ClientPlayLowHealthImpact, ClientEnableLowHealthAudio, and ClientDisableLowHealthAudio, plus MIX_LowHealth class entries used to shape the low-health state.*

6.3 Fall pain and death-event separation

Fall pain and death were treated as separate audio jobs. Fall pain belongs on the damaging-fall branch and is a local player presentation event. Death should immediately clear the low-health state and use a short, final cue rather than reusing a long low-health one-shot.

Backup: *See Appendix I for the damaging-fall branch, owning-client fall-pain event and death-state cue trigger evidence if implemented.*

7. Reverb and spatial presentation

The reverb system was designed as a scalable project-level presentation layer, not a bespoke per-asset reverb solution. The goal was to make interior/exterior transitions demonstrable while keeping the routing simple.

7.1 Shared reverb path

SM_Rvb_Global acts as the shared reverb send/return path. Sounds that should feed the reverb use appropriate sends into this path. The default effect chain represents the exterior space.

Backup: *See Appendix J for the SM_Rvb_Global effect chain, representative MetaSound send routing into SM_Rvb_Global.*

7.2 Interior override logic

Interior regions override the shared reverb effect chain to an interior convolution chain. The preferred implementation uses box collision components inside BP_Building-style actors with overlap-count logic, so adjacent boxes do not cause rapid reverb popping.

Step	Blueprint behaviour	Reason
Begin overlap	Increment InteriorOverlapCount; if count becomes 1, call Set Submix Effect Chain Override.	Switch to interior only on first entry.
Inside region	Leave override active while count > 0.	Avoids popping while crossing adjacent boxes.
End overlap	Decrement using MAX(0, count - 1).	Prevents negative overlap count.
Final exit	If count == 0, clear the override.	Returns to the exterior default chain.
Transition	Use a short crossfade.	Keeps the change audible but smooth.

Backup: *See Appendix J for the BP_Building interior overlap graph, InteriorOverlapCount logic, Set Submix Effect Chain Override / Clear Submix Effect Chain Override nodes, and screenshots of the exterior and interior reverb presets.*

7.3 Spatial and attenuation principles

The project uses attenuation and occlusion as presentation tools, not global fixes. Close props are kept nearfield; world weapons use distance and air absorption; local-only confirmations avoid spatialization where positional information would reduce clarity.

8. Routing, dynamic mixing, and output chain

The final mix architecture is organized around Sound Classes, stem Submixes, and SoundMix assets. The system is intentionally readable: category routing is visible, dynamic behaviour is limited to the events that need it, and final output processing is used for safety rather than loudness abuse.

8.1 SoundClass hierarchy

SClass_Master branches into the main categories used for SFX, UI, and Music. SFX then contains weapons, ambient, footsteps, impacts, explosions, voice/dialog, and related child classes. Extra child classes are used where dynamic-mix behaviour must apply selectively.

Class / child class	Purpose
SClass_Weapons	General weapon and handling content.
SClass_Gunshots_Local	Local FPP gunshots only; drives combat-focus ducking.
SClass_Ambient	Ambience beds, traffic, environmental loops, and prop ambience.
SClass_Footsteps	Footsteps, movement, and landing where routed.
SClass_Impacts / SClass_Explosions	Material impacts and explosive events.
SClass_UI	Routine UI/interface feedback.
SClass_UI_Priority	Objective, countdown, result, or other priority interface events.
SClass_Music / SClass_Music_Stings	Menu music and transition stingers.

Backup: *See Appendix K for the SoundClass graph showing the main project classes and selective child classes used for routing and dynamic mix control.*

8.2 Submix routing

Submix	Content	Purpose
SM_Final	All main stems and reverb return	Final mix point and output safety processing.
SM_Weapons	Gunshots, reloads, handling where routed	Combat stem monitoring/control.
SM_Footsteps	Footsteps and movement detail	Movement stem monitoring/control.

SM_Impacts	Material impacts	Impact stem monitoring/control.
SM_Explosions	Grenades, claymore, barrel content	Explosion stem monitoring/control.
SM_Ambient	Ambience beds and environmental props	Environmental stem monitoring/control.
SM_UI	Interface and local confirmations	UI stem monitoring/control.
SM_Music	Menu music and stingers	Music stem monitoring/control.
SM_Rvb_Global	Shared reverb send/return	Interior/exterior reverb path.

Backup: *See Appendix K for the SM_Final submix graph and Audio Insights Audio Meters showing active stems during gameplay.*

8.3 SoundMix strategy

SoundMix	Trigger method	What it changes	Design reason
MIX_Duck_Environment	Passive modifier on SClass_Gunshots_Local	Gently reduces ambient/background space during local gunfire.	Clears combat space without compressing the whole mix.
MIX_LowHealth	Blueprint push/pop from health state	Class-specific volume and LPF changes while critical.	Represents player state while preserving gameplay information.
MIX_UI_Priority	Priority child classes or targeted Blueprint call	Ducks background space for objective/countdown/result events.	Lets important UI read without making all UI louder.
MIX_FPS_SoundMix	Menu/user settings path	General user-facing class volume behaviour.	Kept separate from gameplay ducking.

8.4 Final output safety

SM_Final uses output control as a safety layer. The limiter is there to prevent overs and capture-chain issues, not to crush transients or create loudness through heavy gain reduction.

9. Loudness, profiling, and validation workflow

The validation process combines gameplay listening with measurable checks. The project does not treat spreadsheet targets as absolute truth; isolated measurements are used to catch obvious problems, then final decisions are checked in game context.

9.1 Audio Insights and Unreal Insights usage

Audio Insights is useful for live audio-state inspection: active sounds, meters, submix activity, and runtime behaviour. Unreal Insights Timing view is useful for CPU/performance review but is not a direct audio loudness review tool.

9.2 Isolated MetaSound render and WAV analysis

A batch capture workflow was built to render MetaSoundSource assets in isolation to WAV files. Those files were then analysed with a Python script for duration, sample peak, RMS, 400 ms window value, and longer-window RMS. This revealed hot UI assets and helped sanity-check weapon, explosion, and footstep relationships.

Workflow stage	Purpose
Batch MetaSound render	Create isolated WAVs for each tested MetaSound.
Offline WAV analysis	Measure peak/RMS/momentary-style values outside the engine.
Spreadsheet review	Compare relative asset families and flag obvious outliers.
In-game review	Confirm whether measured changes actually improve gameplay readability.

9.3 Final gameplay review approach

The final review used one controlled gameplay capture route: menu, loadout, spawn, traversal, outdoor movement, indoor movement, rifle and pistol fire, impacts/explosions, UI/objective events, low-health if available, and return to neutral ambience.

Review item	Practical target / judgement
Integrated loudness	Useful range for gameplay capture is roughly -22 to -24 LUFS, not a broadcast master target.
True peak	Keep below 0 dBTP, with practical safety around -1 to -2 dBTP.
Limiter behaviour	No obvious pumping, crushed weapons, or flattened ambience.
Combat readability	Weapons, confirmations, and impacts remain clear while ambience still exists.

Dynamic mixing	Ducking should support focus without being more noticeable than the action itself.

10. Asset management, audit, repair, and cleanup workflow

The project included a substantial maintenance pass. This matters because technical audio work is not only asset creation; it also requires reference integrity, naming discipline, routing consistency, and confidence that old content is not silently driving the build.

10.1 Audio audit workflow

Unreal Python audit workflow exported audio inventory, naming suggestions, routing assignments, referencer relationships, SoundClass/Submix trees, SoundMix summaries, and probably-unused candidates. It provided a concrete baseline before cleanup and mixing decisions.

> **Backup:** *See Appendix B and the attached Vector_Breach_Audio_Audit_Appendix_Workbook.xlsx for the full audio audit summary, asset relationship index, SoundClass/Submix tree exports, routing review flags, and probably-unused asset candidates.*

10.2 Broken reference and rename repair

After rename and move work, some MetaSounds still serialized references to old attenuation and concurrency packages. A repair script was created to rebind known-broken references, and a critical disconnected-asset report was used to identify high-value assets that had lost callers.

10.3 Tooling deliverables

- ExportAudioAudit and related audio inventory reports.
- Purge / cleanup scripts for unused cues and unreferenced SoundWaves.
- Broken attenuation/concurrency reference repair script.
- Critical disconnected-asset reporting.
- SoundWave ownership / folder / naming reports.
- MetaSound batch capture script.
- Offline WAV analyzer script and output spreadsheet.

Appendix A. Major system and asset index

Group	Assets / systems to evidence
Environment	BP_AmbienceManager, MSP_AmbienceState, MS_Amb_Bed_Outdoor, MS_Traffic_Perimeter, BP_TrafficEmitter, BP_BushEmitter, BP_ConsoleDeskEmitter, BP_LightHumEmitter, BP_FanEmitter, BP_LargeFanEmitter, MS_ElectricBox, MS_Flag.
Weapons	MS_WPN_AR_Shot_FPP, MS_WPN_AR_Shot_TPP, MS_WPN_Pistol_Shot_FPP, MS_WPN_Pistol_Shot_TPP, reload MetaSounds, bullet cases, dry fire, zoom.
Movement/Foley	MS_Footstep_Generic, MS_Footstep_Wood, MS_Footstep_Metal, MS_Footstep_Glass, MS_FoleyBody, MS_Foley_Body_Move, MS_Foley_HandSwipe, MS_Land_Generic, MS_Land_Wood, MS_Land_Metal, MS_Land_Glass.
Impacts/Explosions	MS_Impact_Default, MS_Impact_Wood, MS_Impact_Metal, MS_Impact_Glass, MS_Impact_Body_World, MS_Explosion_Barrel, MS_Explosion_Claymore, MS_Explosion_Grenade.
UI/State	MS_UI_Hitmarker, MS_UI_Hitmarker_Headshot, MS_UI_Objective_Notification, MS_UI_Menu_Highlight, MS_UI_LowHealth, restock confirm, MS_FallPain, death cue.
Routing/Mix	SClass_Master, SClass_SFX, SClass_Weapons, SClass_Gunshots_Local, SClass_UI, SClass_UI_Priority, SClass_Music_Stings, SM_Final, SM_Weapons, SM_Ambient, SM_UI, SM_Rvb_Global, MIX_Duck_Environment, MIX_LowHealth, MIX_UI_Priority, SMFX_Final_Limiter.
Tooling	ExportAudioAudit, purge/cleanup scripts, broken reference repair, SoundWave ownership reports, MetaSound batch capture script, WAV analyzer script.

Appendix B. Dashboard asset summary

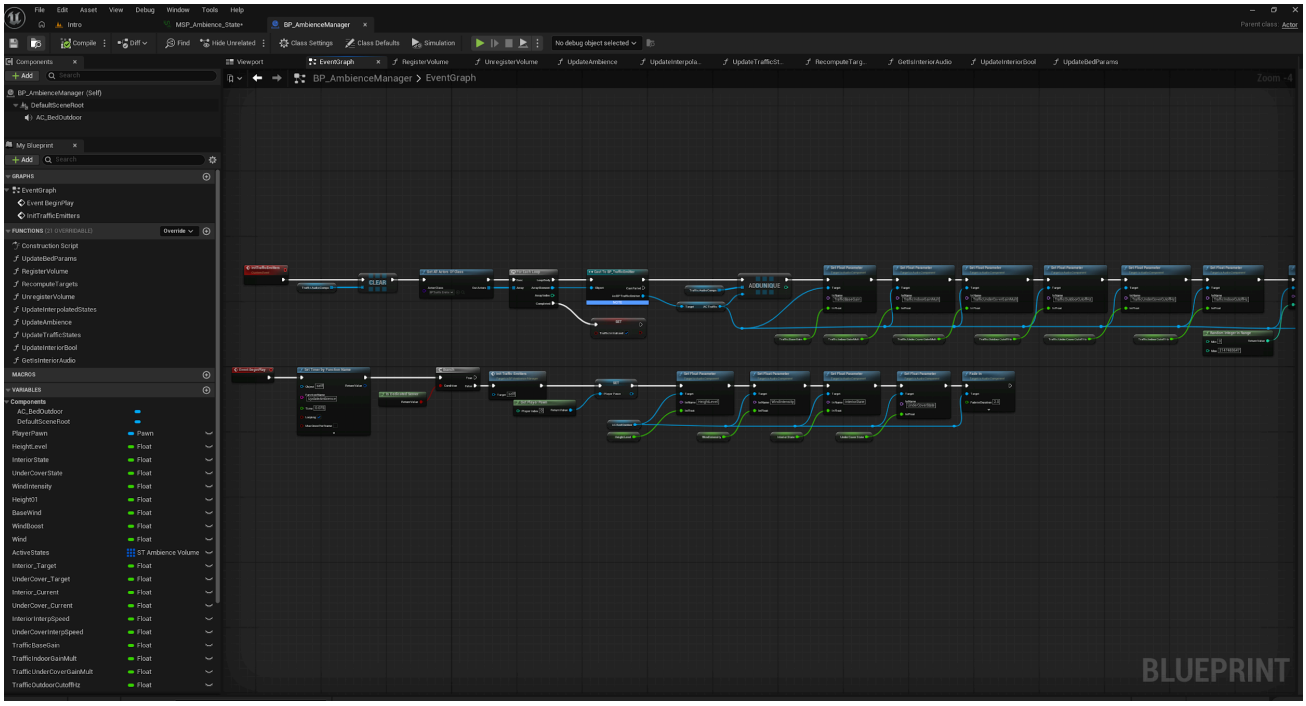
Vector Breach_Audio_Audit

Caption:

Audit dashboard used to support the asset-management section. The companion workbook Vector_Breach_Audio_Audit_Appendix_Workbook.xlsx contains the full asset relationship index, references, SoundClass/Submix exports, routing issues, and probably-unused candidates.

Appendix C — Ambience State System Evidence

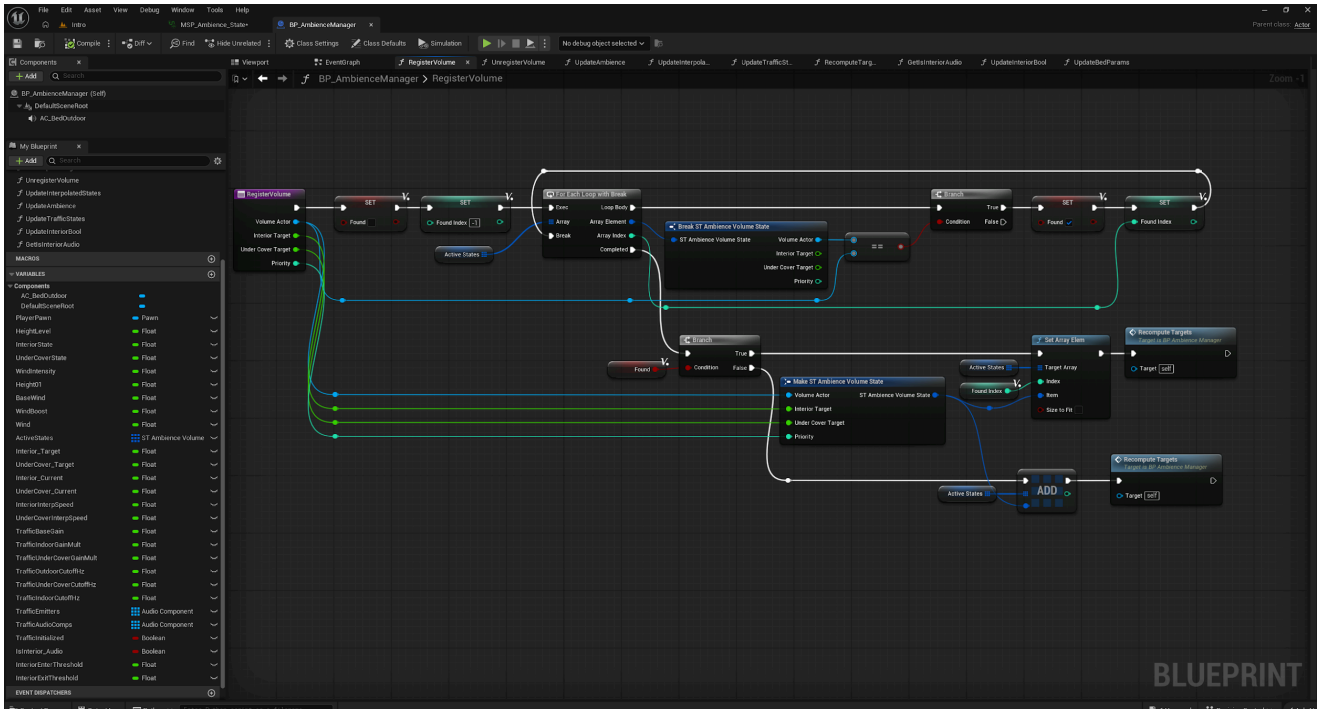
C.1 BP_AmbienceManager, EventGraph Overview



Caption:

Full overview of the ambience manager update flow. This Blueprint gathers runtime ambience state and pushes smoothed interior, under-cover, wind, and gain parameters into the ambience audio system.

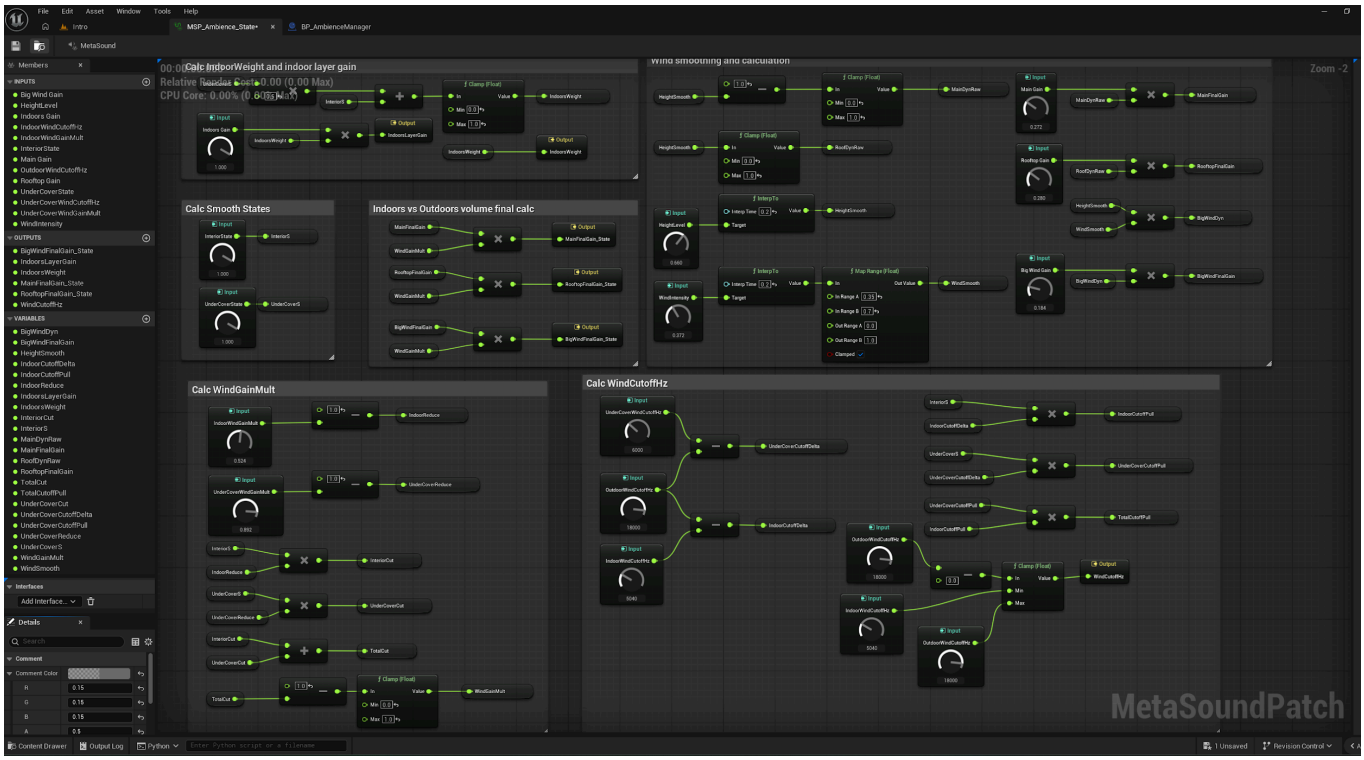
C.2 BP_AmbienceManager — RegisterVolume Function



Caption:

Supporting function used to register ambience volumes into the manager’s active-state array. This allows interior and under-cover spaces to contribute to the current ambience state.

C.3 MSP_AmbienceState — Full Patch Overview

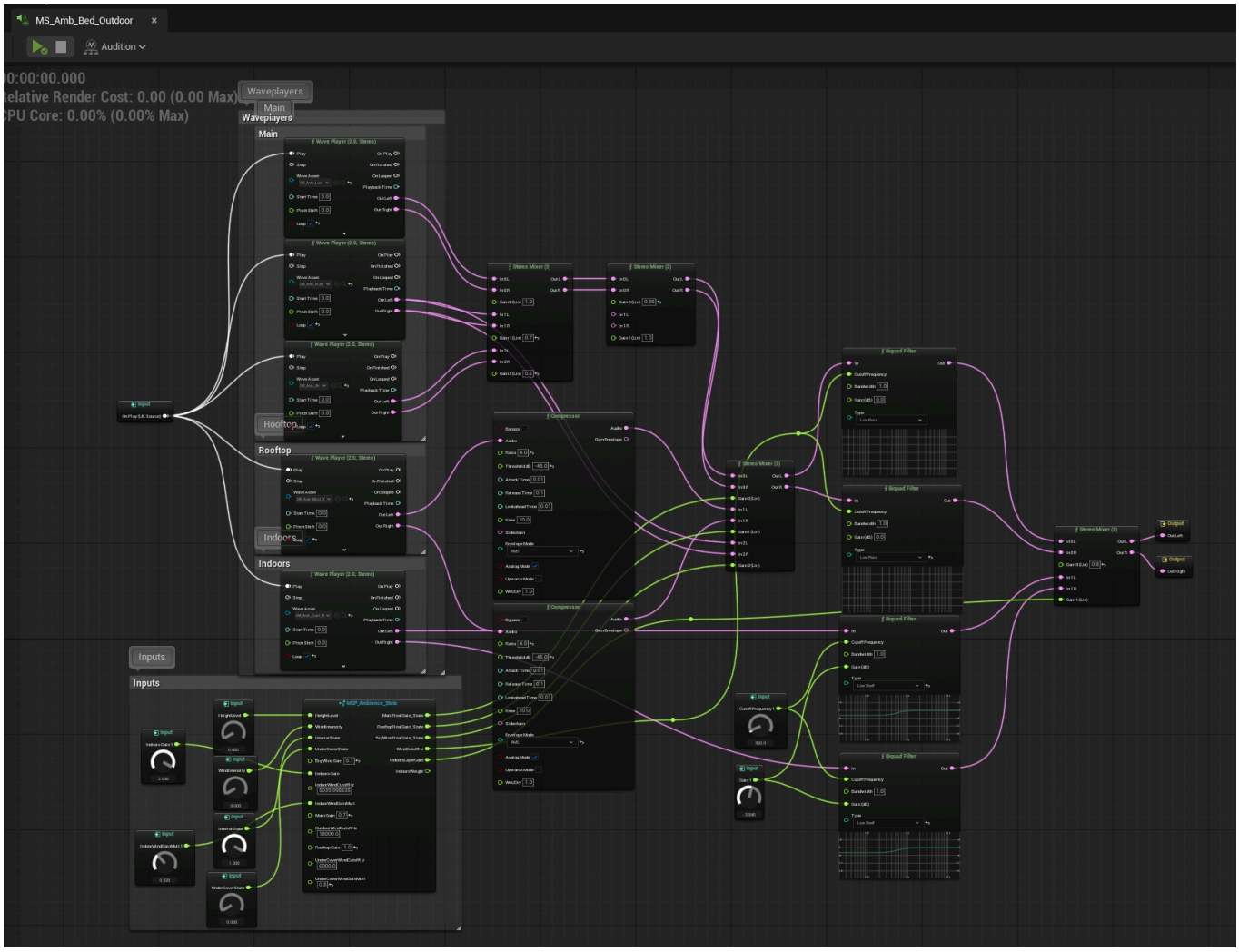


Caption:

Reusable MetaSoundPatch used to calculate indoor weighting, indoor layer gain, wind reduction, rooftop/main wind balance, and cutoff frequency behaviour.

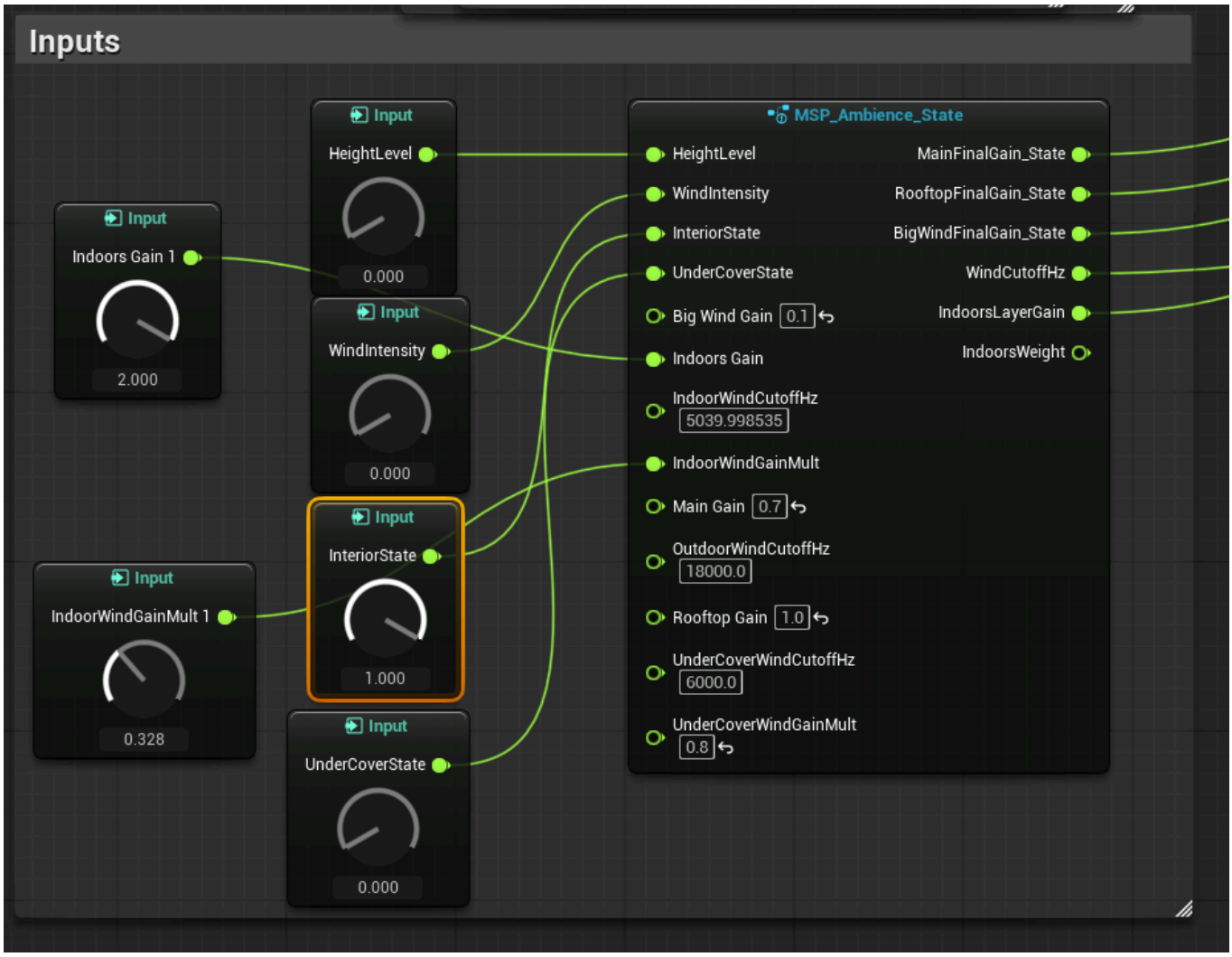
Appendix D — Outdoor Bed and Traffic Ambience Evidence

D.1 MS_Amb_Bed_Outdoor — Full Graph Overview



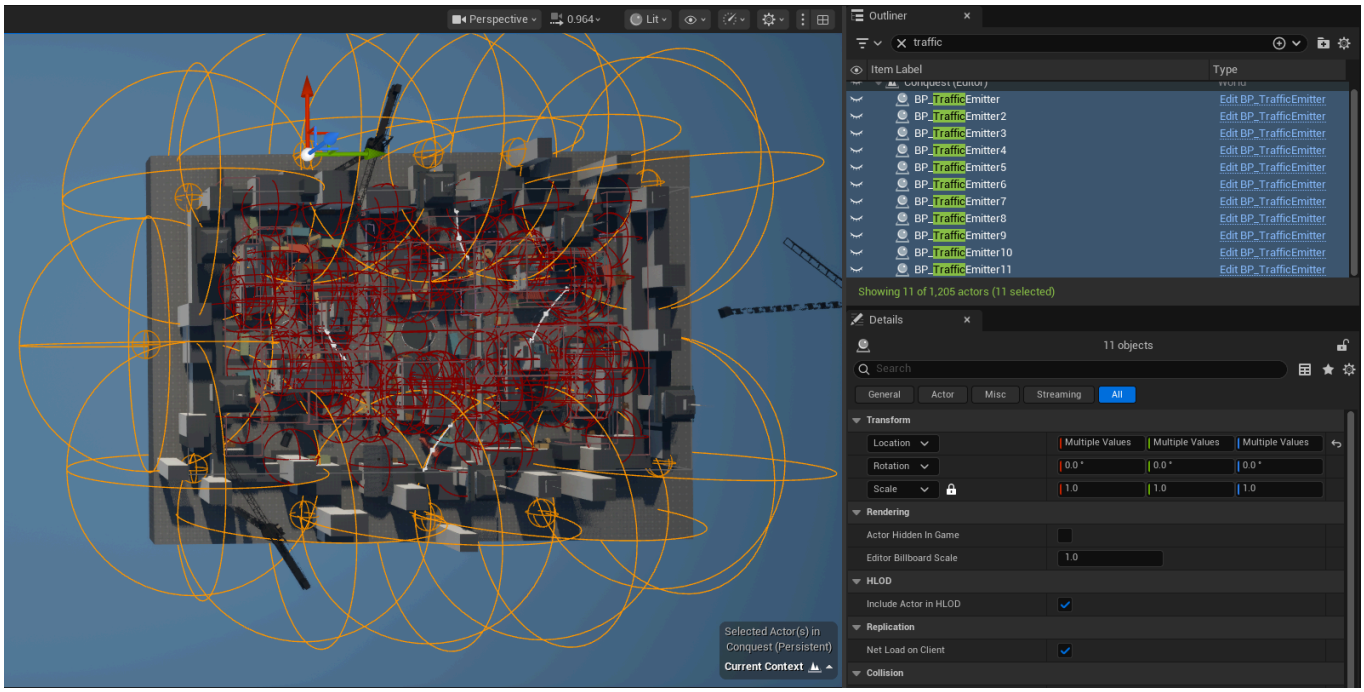
Caption:
Full overview of the outdoor ambience MetaSound. MS_Amb_Bed_Outdoor blends the main exterior bed, rooftop / height-based wind layers, larger wind elements, and indoor contribution while receiving smoothed state values from the ambience system.

D.2 MS_Amb_Bed_Outdoor — Layer Blend / Parameter Callout



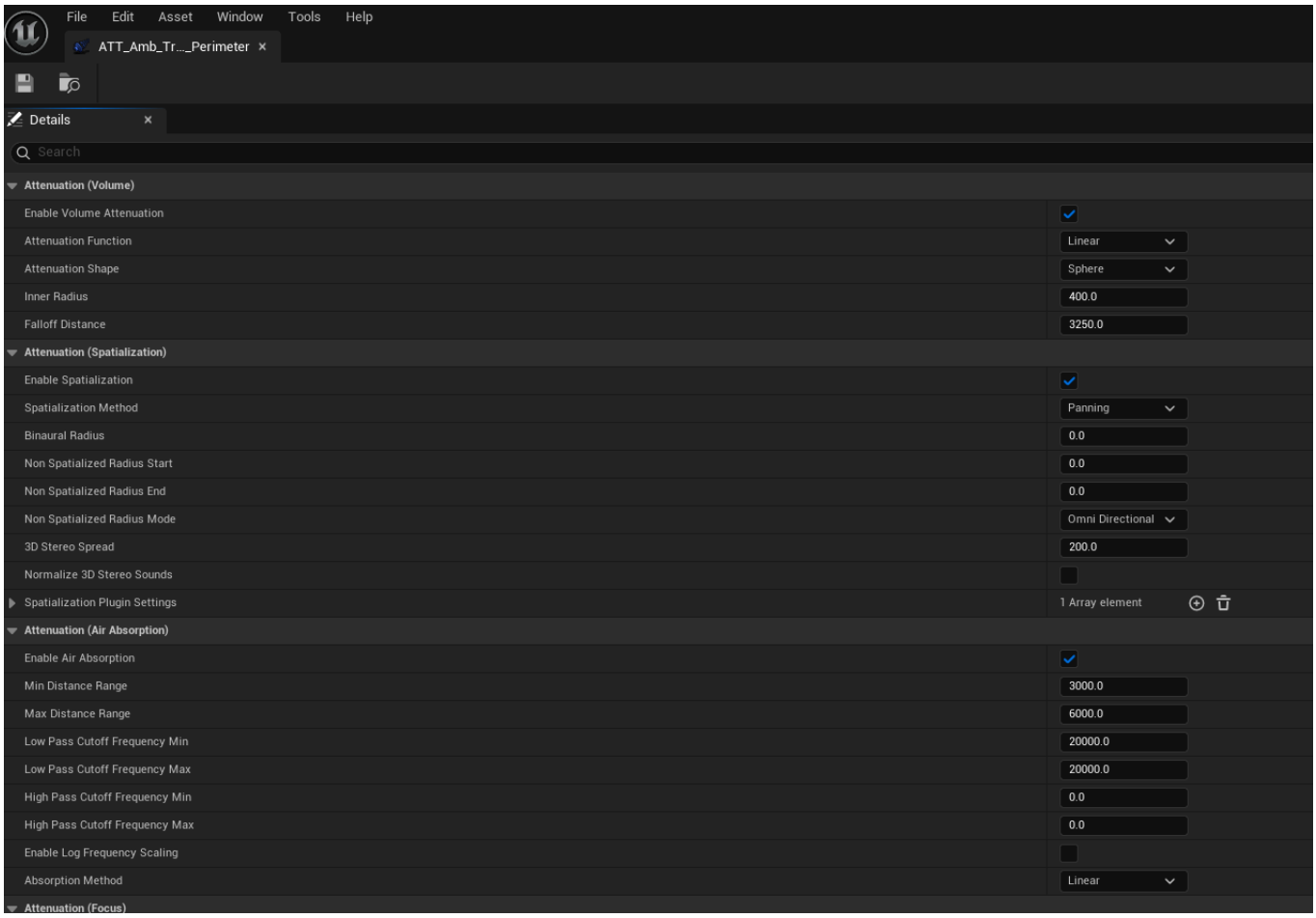
Caption:
Layer and parameter section of MS_Amb_Bed_Outdoor. Runtime parameters from BP_AmbienceManager and MSP_AmbienceState control the balance between outdoor wind, rooftop wind, big wind, filtered ambience, and indoor contribution.

D.3 BP_TrafficEmitter — Level Placement Examples



Caption:
Traffic emitters placed around the level perimeter to create a low-level exterior city / road presence. The system uses spatial placement rather than a single global loop so the traffic bed has direction and distance behaviour.

D.4 Traffic Attenuation Settings



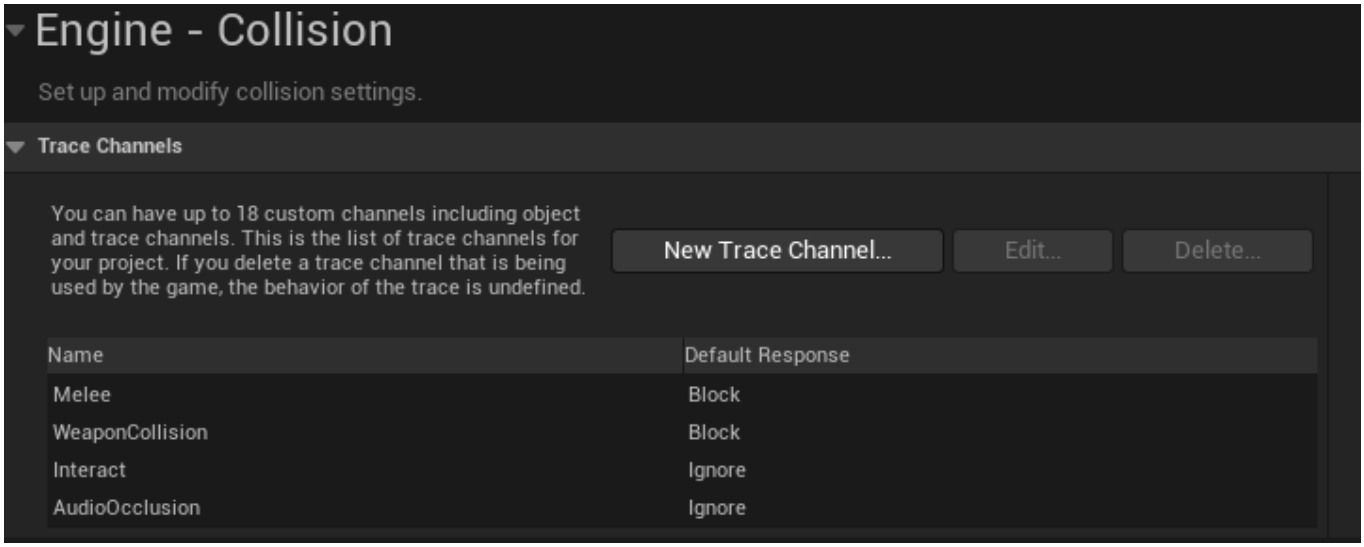
Caption:
Traffic attenuation settings used to keep perimeter traffic broad and environmental rather than close or point-like. The attenuation shape, falloff, filtering help the traffic sit behind the main ambience and gameplay mix.

D.5 Audio Insights — SM_Ambient Runtime Activity

Caption:
Audio Insights runtime view showing SM_Ambient activity. This verifies that the ambience bed and traffic system route through the intended ambient stem for monitoring and mix control.

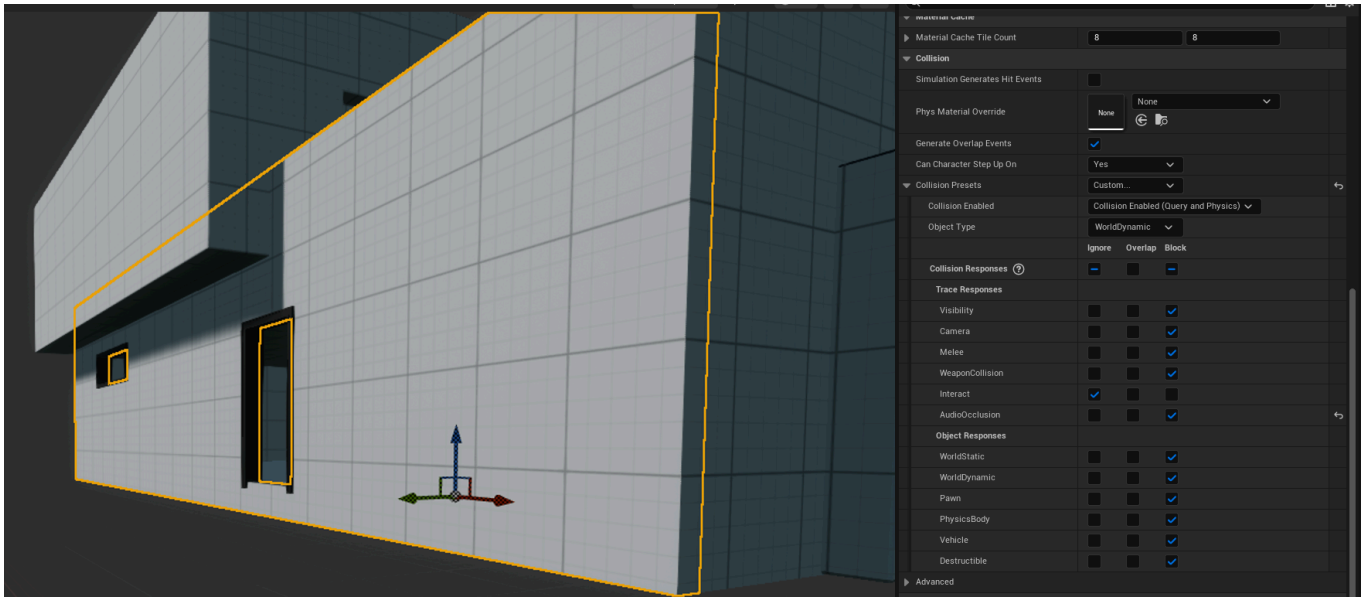
Appendix E — AudioOcclusion and Collision Evidence

E.1 Project Settings — AudioOcclusion Trace Channel



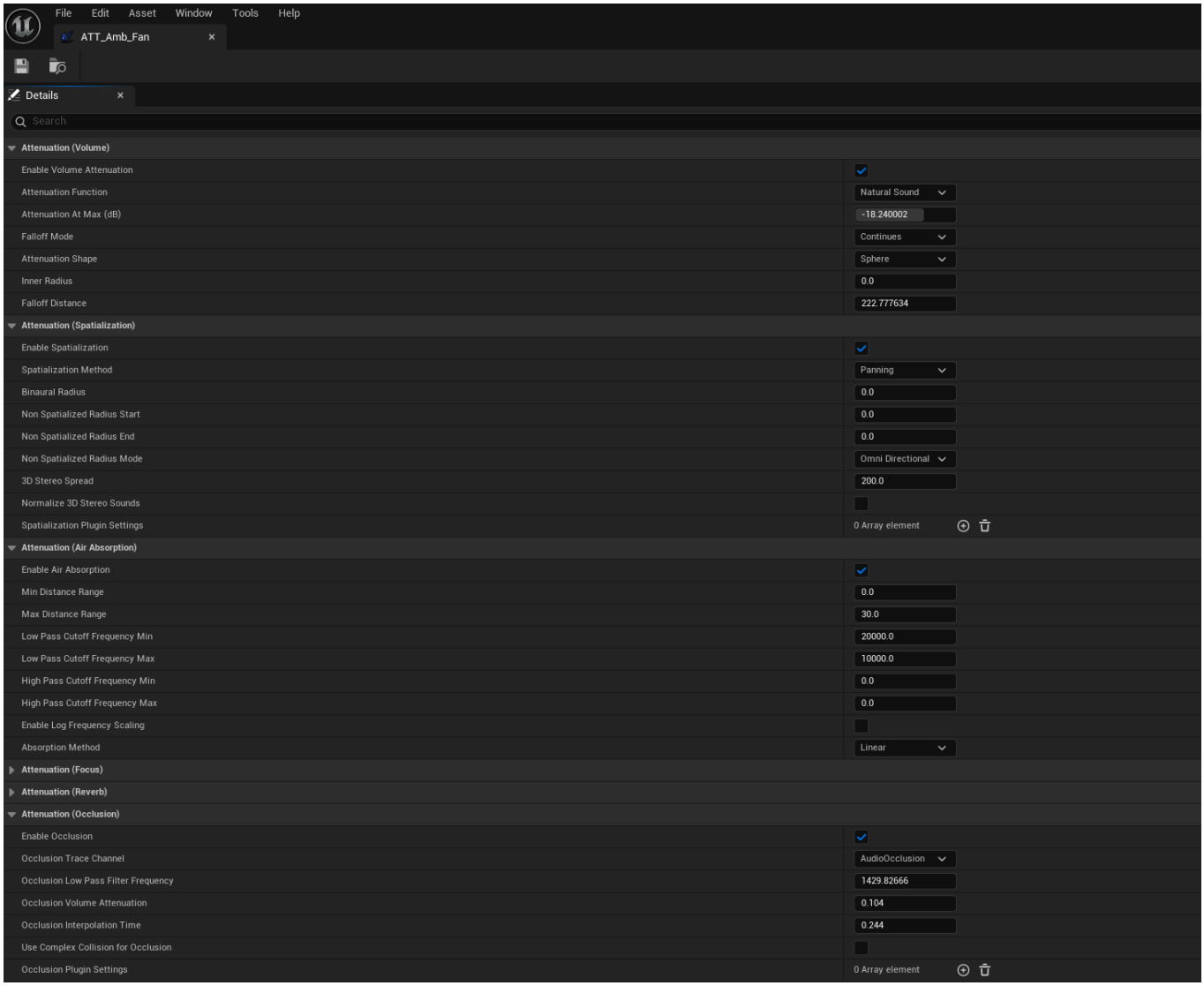
Caption:
Project collision settings showing the custom AudioOcclusion trace channel. This channel was created so audio muffling / occlusion could be handled separately from Visibility and gameplay traces.

E.2 Structural Wall Mesh — AudioOcclusion Response



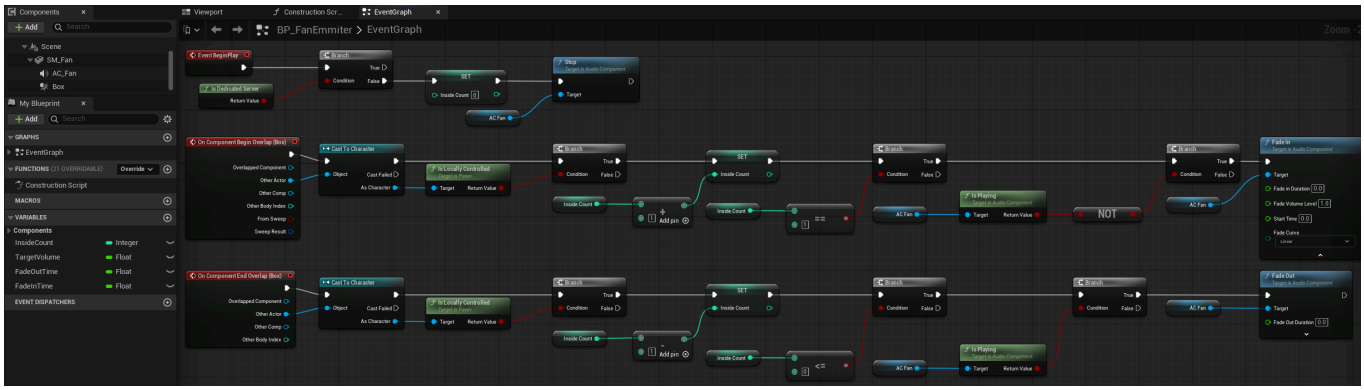
Caption:
Example structural wall collision response. Walls block AudioOcclusion, allowing attenuation occlusion traces to detect indoor / outdoor separation without relying on the gameplay Visibility channel.

E.3 Attenuation Asset — Occlusion Trace Channel



Caption:
Representative attenuation settings using AudioOcclusion as the occlusion trace channel. This allows environmental sounds behind walls to be muffled without affecting weapon traces, camera traces, or other gameplay systems.

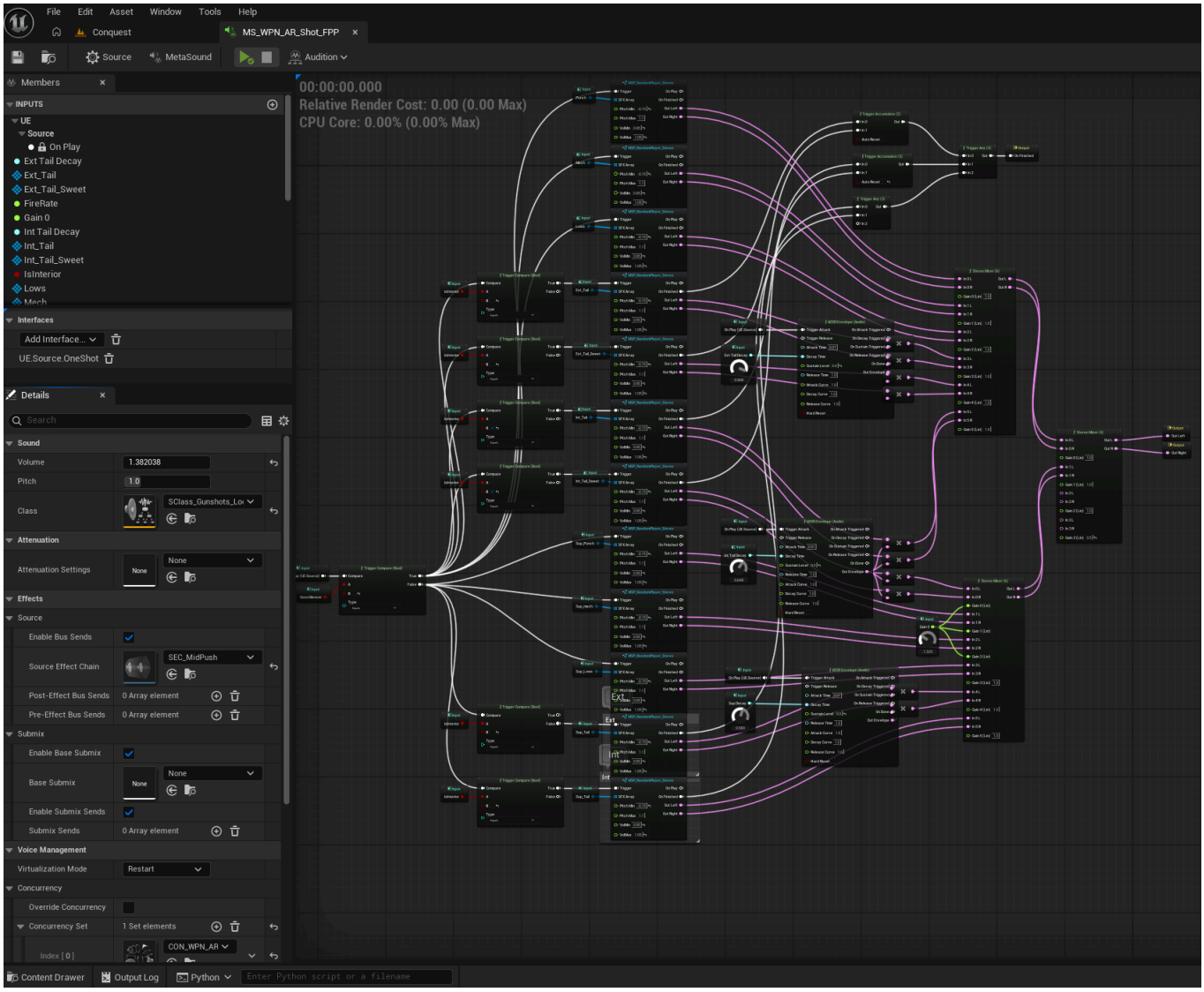
E.4 Example Emitter Collision Settings



Caption:
Representative audio emitter collision setup. Audio trigger volumes are query-only overlap volumes that detect the player pawn while ignoring gameplay traces, preventing audio spheres / boxes from interfering with bullets, camera, or interaction logic.

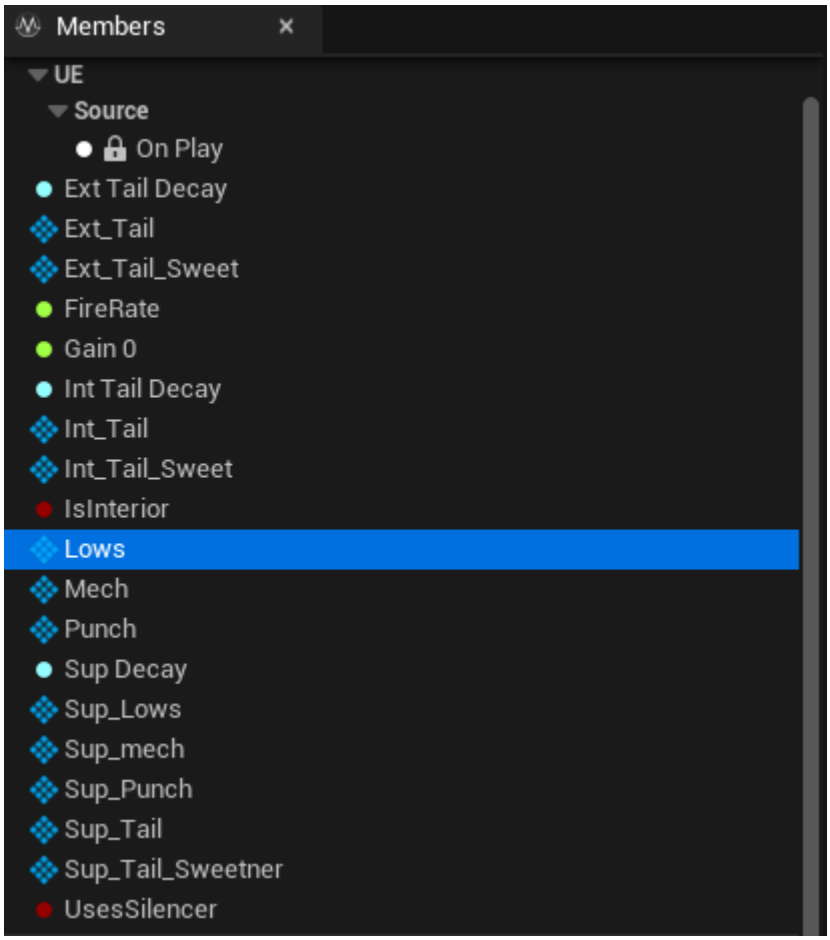
Appendix F — Assault Rifle Shot System Evidence

F.1 MS_WPN_AR_Shot_FPP — Full Graph Overview



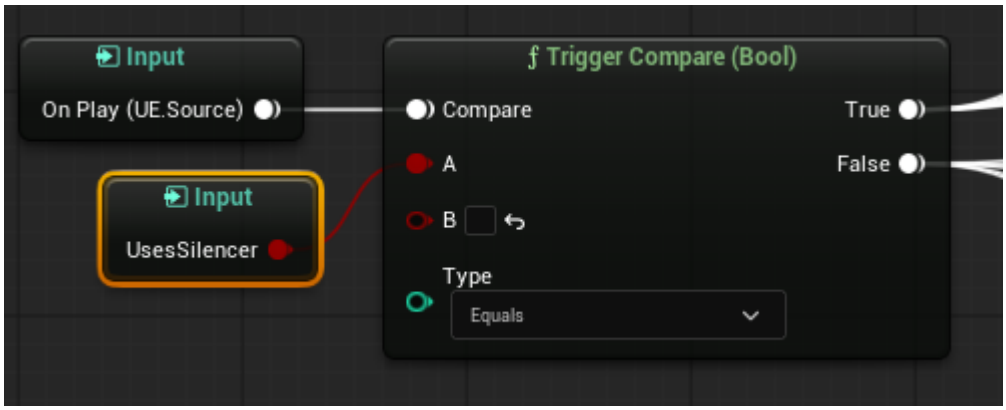
Caption:
Full overview of the first-person assault rifle shot MetaSound. The graph handles layered one-shot playback, suppressed / unsuppressed logic, interior / exterior tail behaviour, and final output balancing.

F.2 MS_WPN_AR_Shot_FPP — Layer Arrays



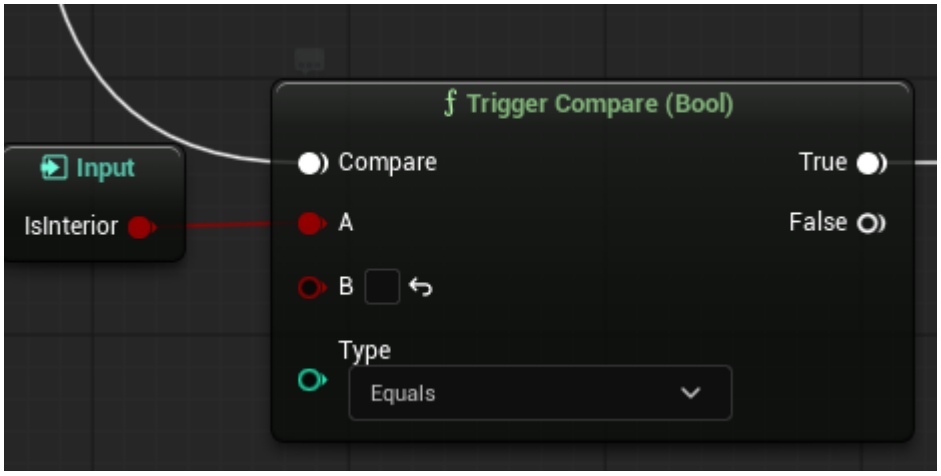
Caption:
Layer-array section of MS_WPN_AR_Shot_FPP. Separate arrays are used for the main shot body, mechanical detail, and tail content so the weapon can be built from multiple controllable layers rather than a single baked gunshot.

F.3 MS_WPN_AR_Shot_FPP — Suppressed Branch



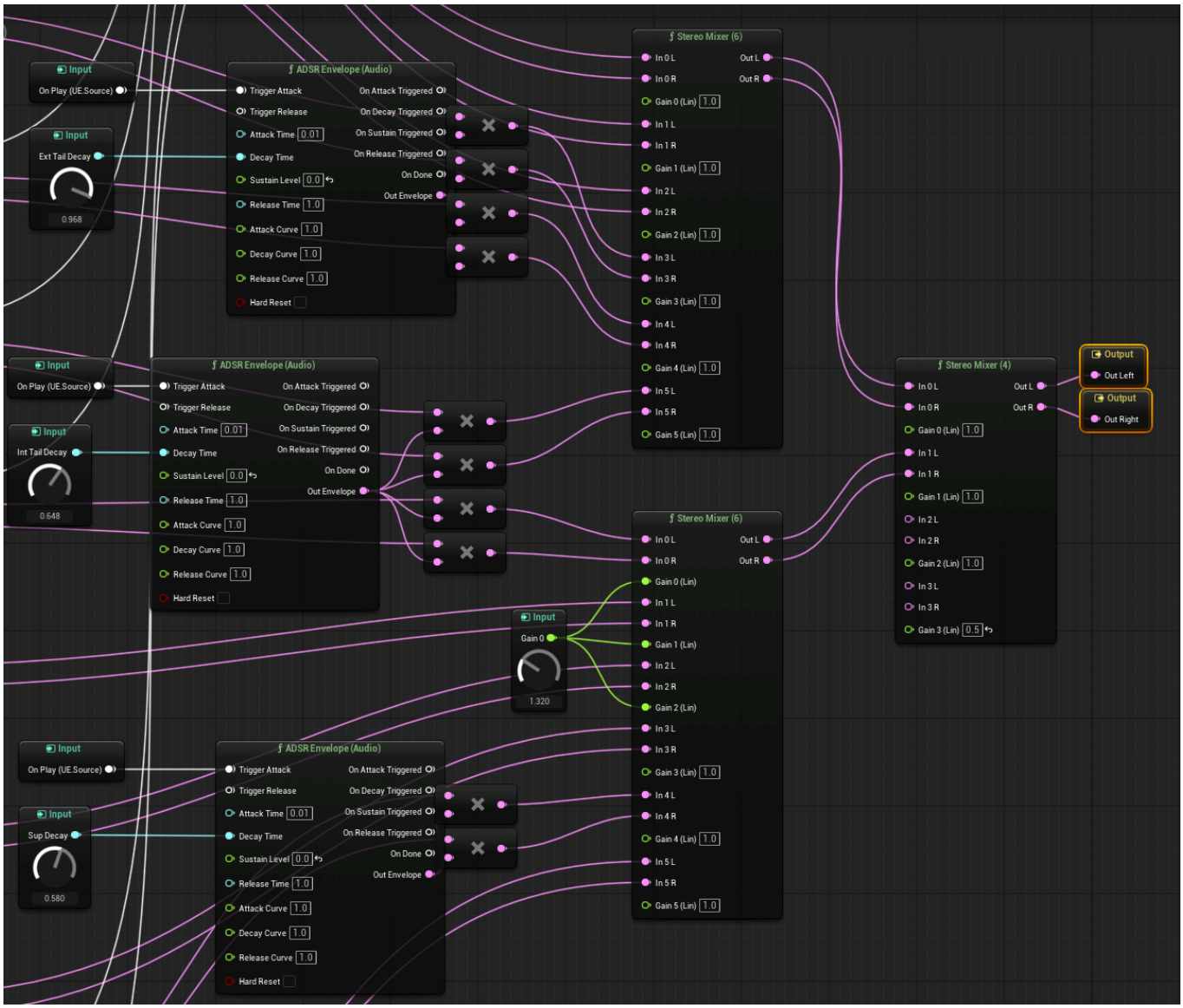
Caption:
Suppressed-branch logic in MS_WPN_AR_Shot_FPP. The graph switches to the suppressed shot path when the weapon's silencer state is active, allowing the same system to support both unsuppressed and suppressed playback.

F.4 MS_WPN_AR_Shot_FPP — IsInterior Tail Selection



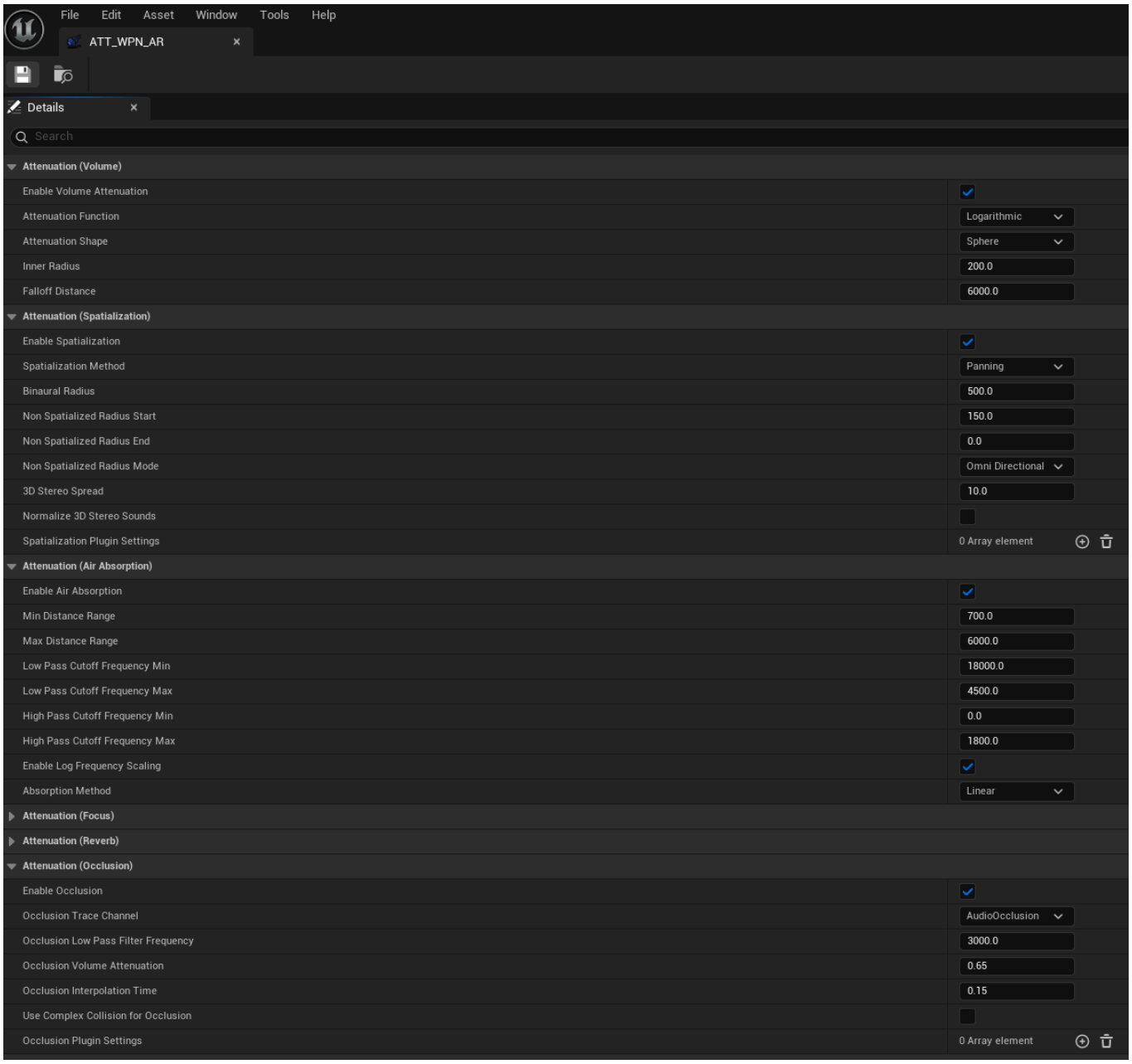
Caption:
Interior / exterior tail-selection logic. The IsInterior state determines which tail path is used, allowing the rifle to swap between tighter indoor decay and broader exterior tail behaviour without duplicating the full weapon graph.

F.5 MS_WPN_AR_Shot_FPP — Output Mixer / Final Output



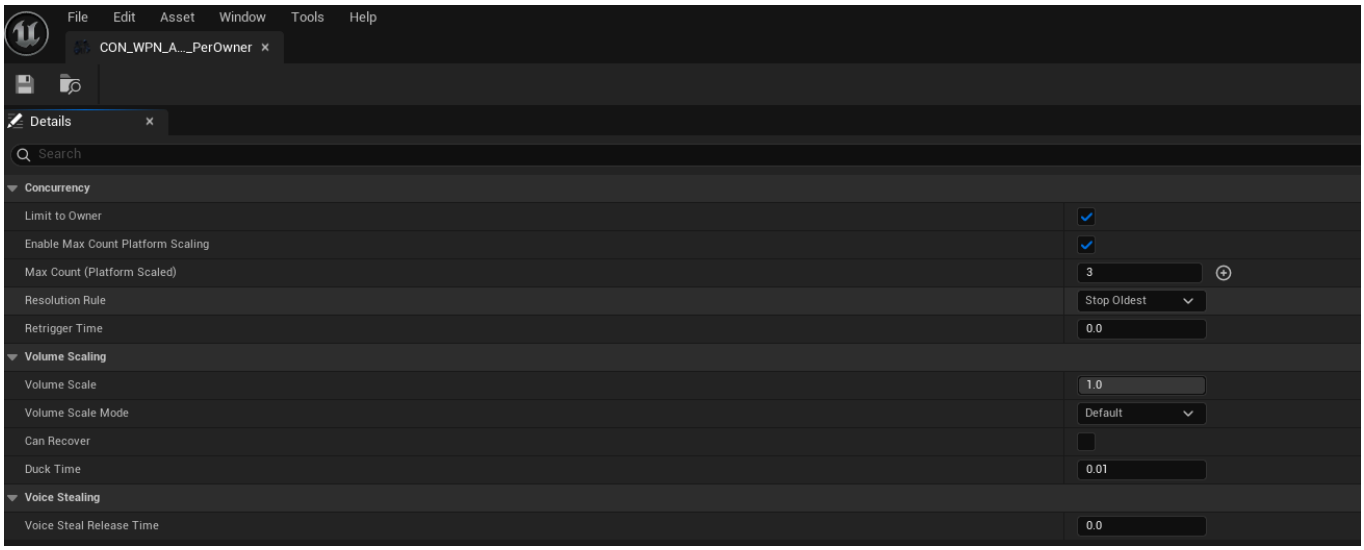
Caption:
Final output-mix section of the FPP rifle MetaSound. Individual layers are balanced before reaching the final output so the overall shot can be tuned as a coherent first-person weapon presentation.

F.6 MS_WPN_AR_Shot_TPP — Attenuation Settings



Caption:
Third-person / world rifle attenuation settings. These settings control range, distance rolloff, filtering, and occlusion so the rifle reads appropriately in-world without behaving like a first-person local shot.

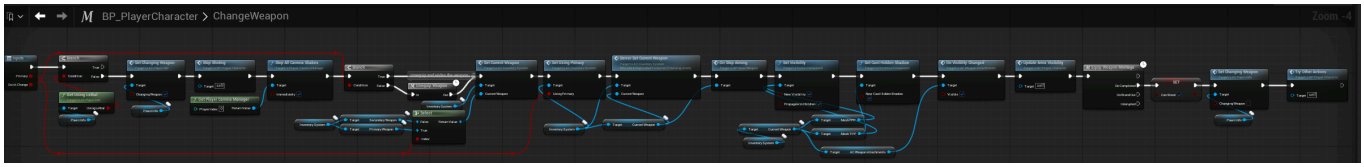
F.7 MS_WPN_AR_Shot_TPP — Concurrency Settings



Caption:
Third-person rifle concurrency settings. Concurrency limits help prevent excessive voice buildup when multiple world weapons are active at once, keeping the combat mix readable.

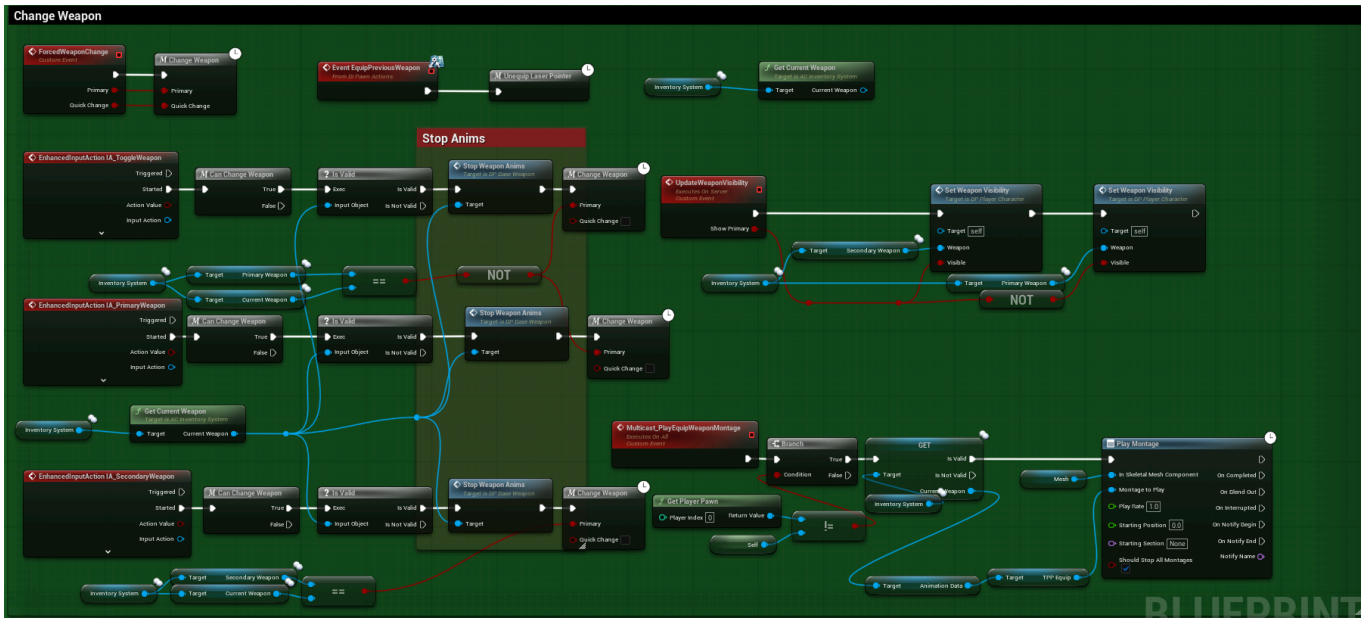
Appendix G — Weapon Swap and Reload Timing Evidence

G.1 BP_PlayerCharacter — Change-Weapon Graph Overview



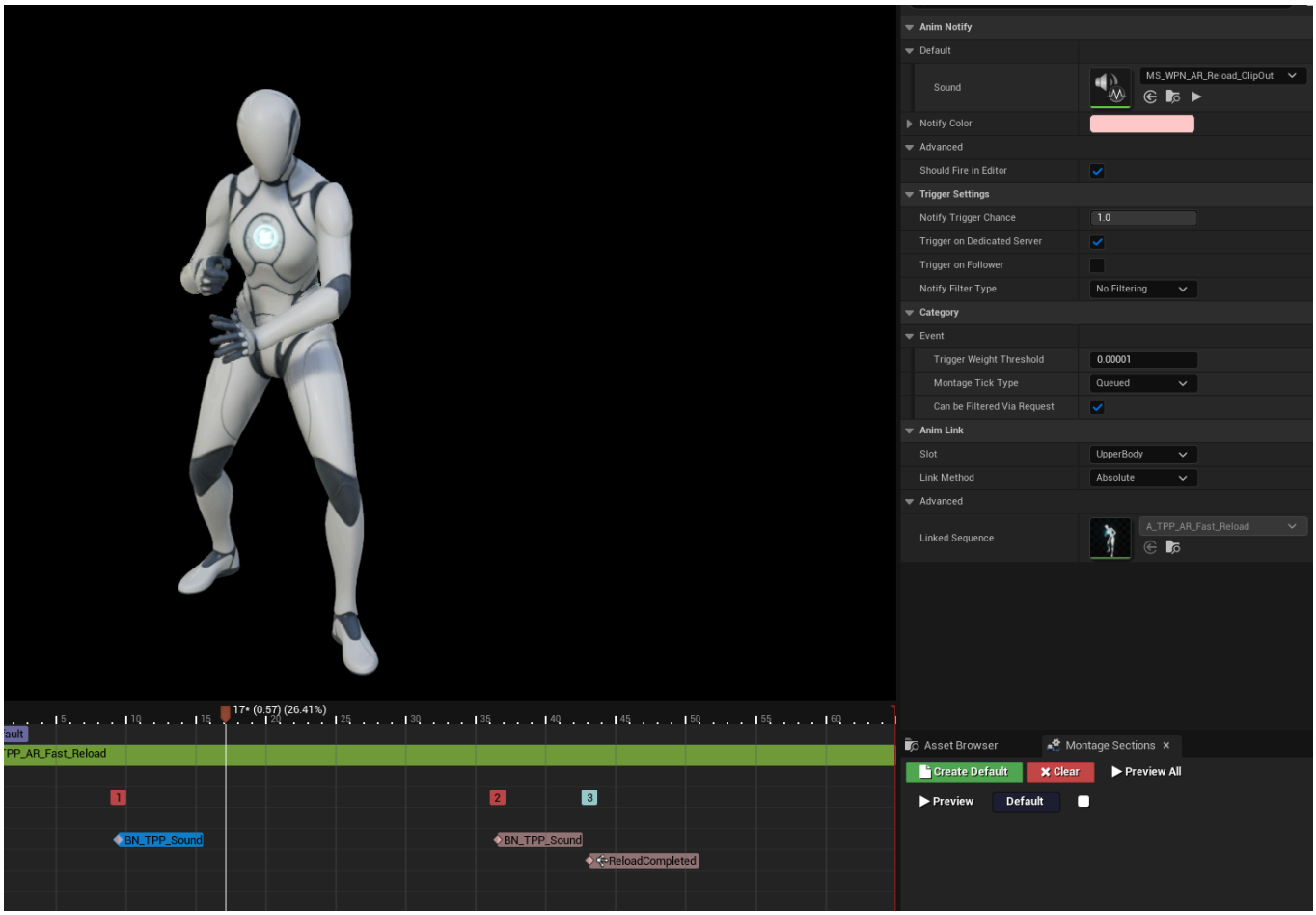
Caption:
Overview of the weapon-change logic in BP_PlayerCharacter. The graph shows the change-weapon path where the current weapon is stopped before the new weapon is equipped.

G.2 BP_PlayerCharacter — StopWeaponAnims Before M ChangeWeapon



Caption:
Close-up of the reload-interruption fix. StopWeaponAnims is called on the current weapon before M ChangeWeapon, preventing delayed reload notifies such as ClipIn from firing after the player has already swapped weapons.

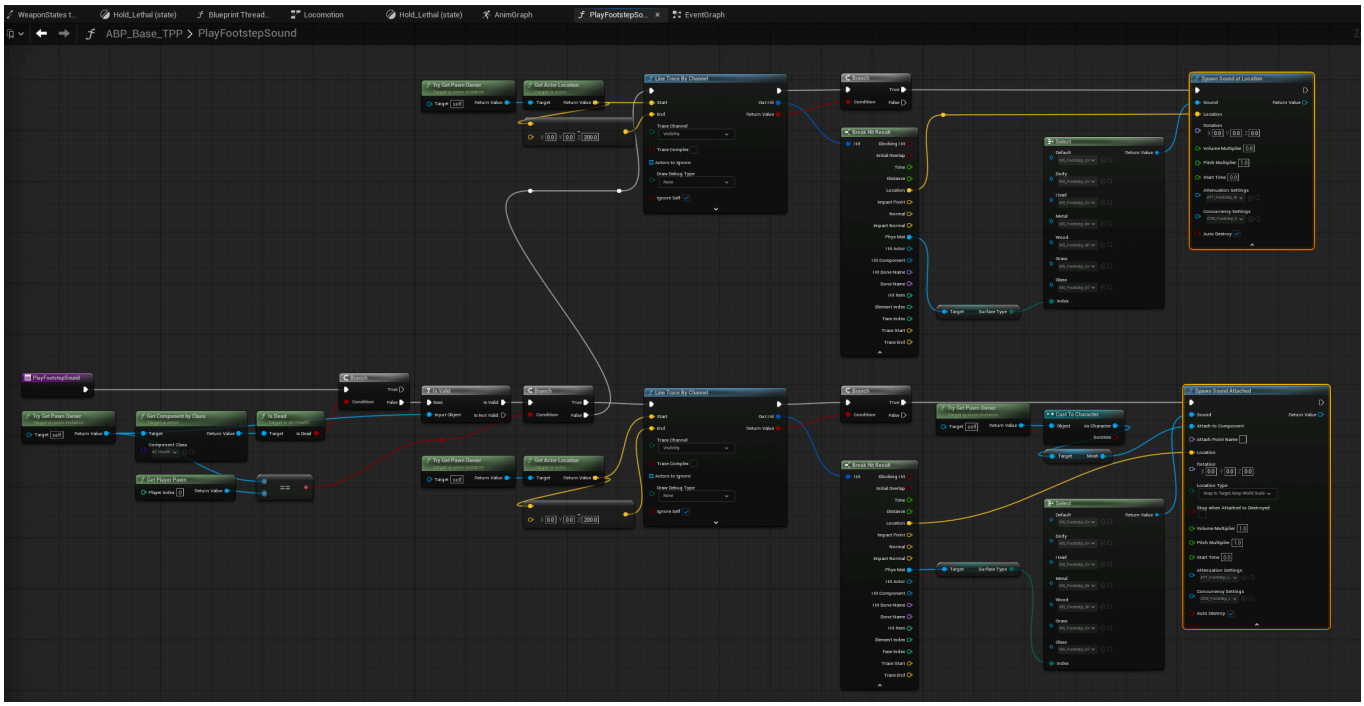
G.3 Reload Animation — ClipOut / ClipIn Notify Timing Overview



Caption:
Reload animation timeline showing the audio notify positions for ClipOut and ClipIn. These animation notifies preserve timing against the weapon animation while routing playback to the replacement MetaSound reload assets.

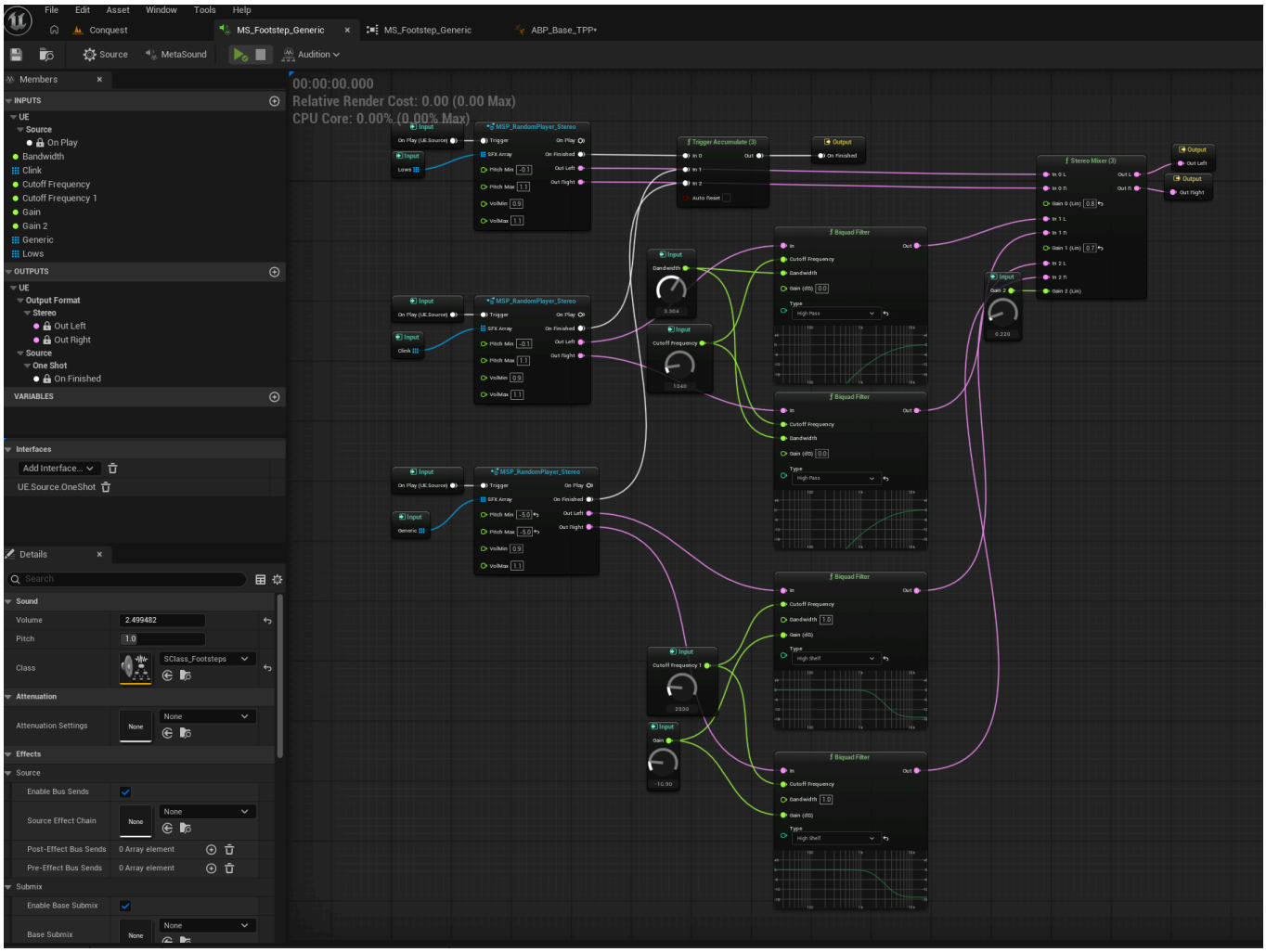
Appendix H — Footstep System Evidence

H.1 ABP_Base_TPP — Surface Material Routing



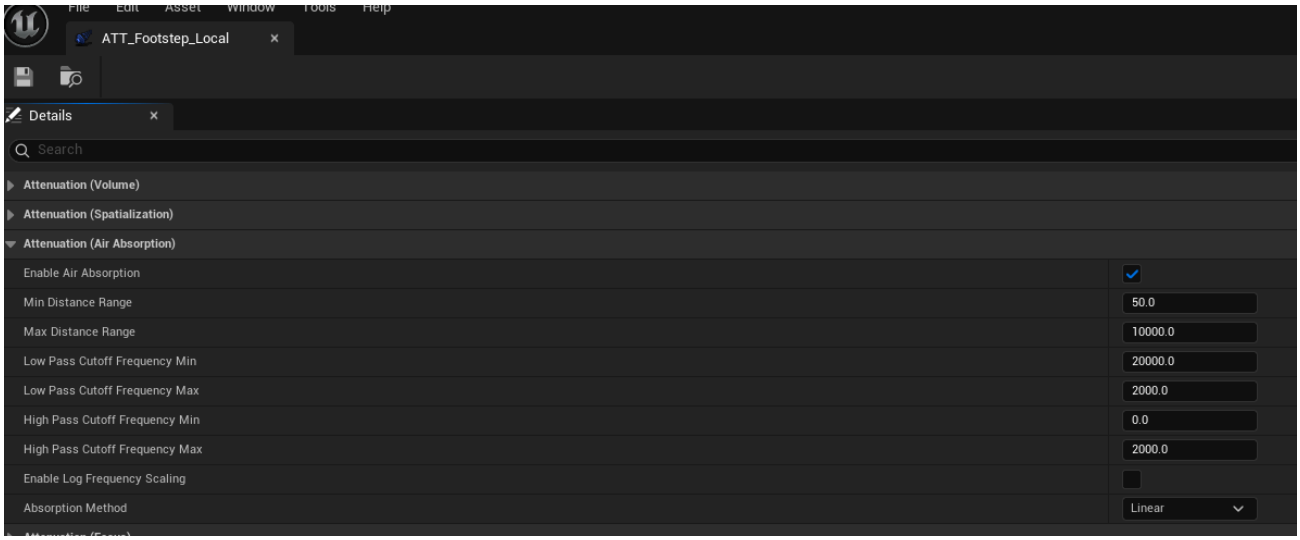
Caption:
Overview of the footstep material-routing logic in ABP_Base / ABP_Base_TPP. The animation Blueprint traces the current surface and routes playback to the correct footstep asset set based on the resolved physical material.

H.2 MS_Footstep_Generic — Array / Playback Overview



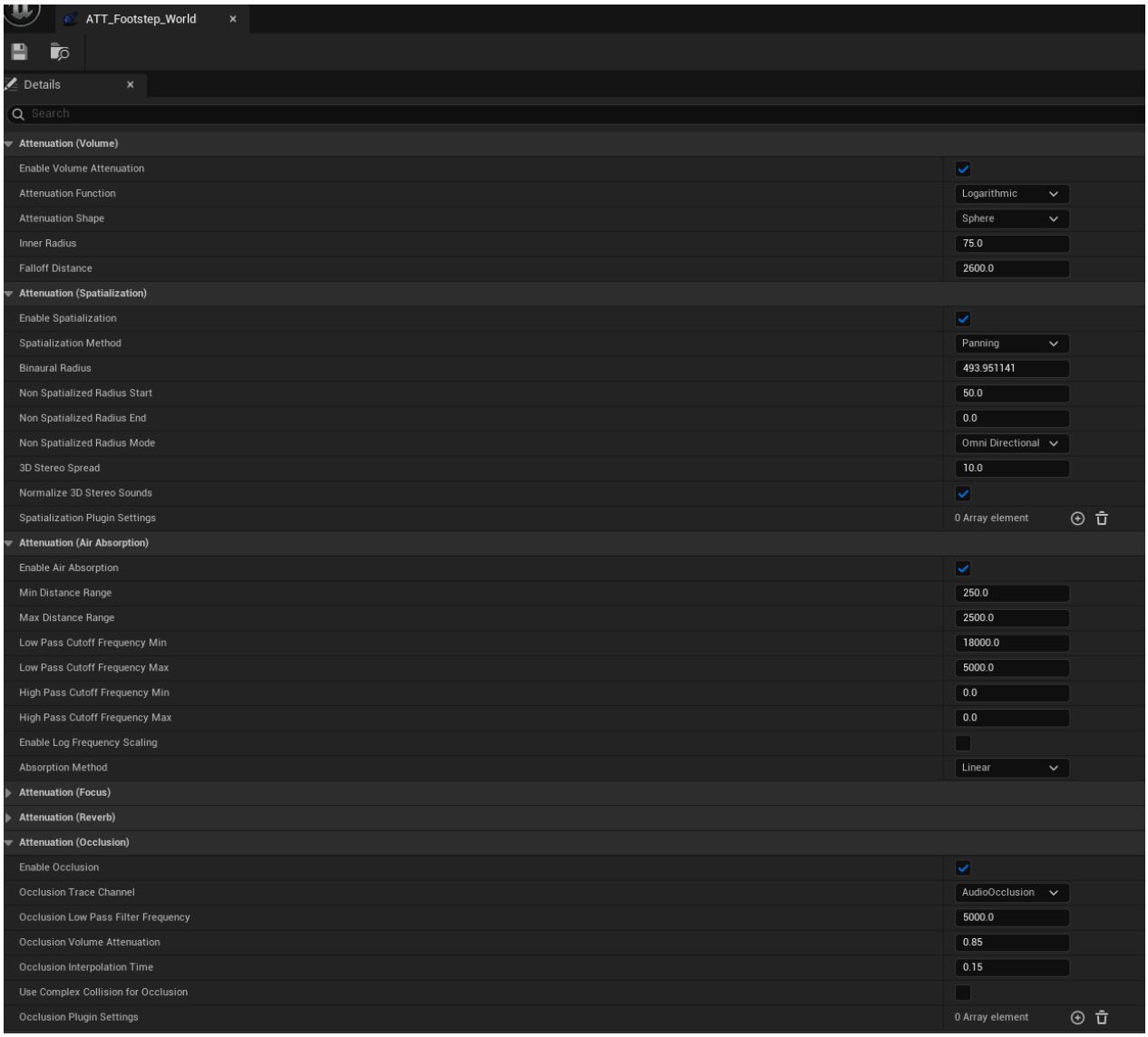
Caption:
Representative array and playback structure for MS_Footstep_Generic. The MetaSound uses array-based one-shot selection to provide variation for default / concrete-style footsteps.

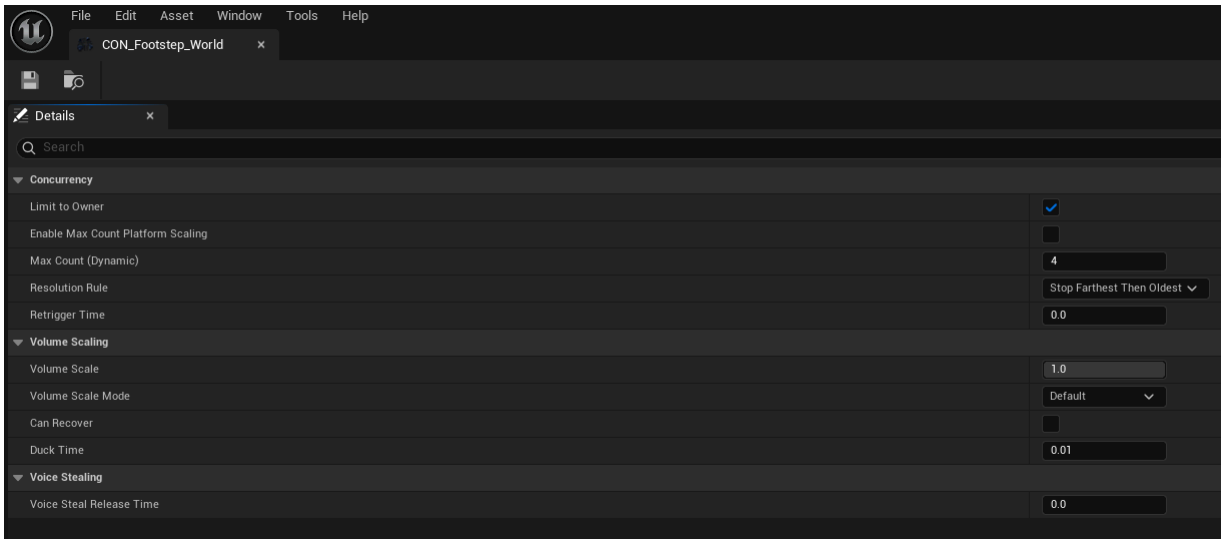
H.3 Local Footstep Attenuation / Concurrency Assets



Caption:
Local-footstep attenuation and concurrency settings. Local player footsteps use their own playback treatment so first-person movement remains immediate and appropriately scaled for self-perspective.

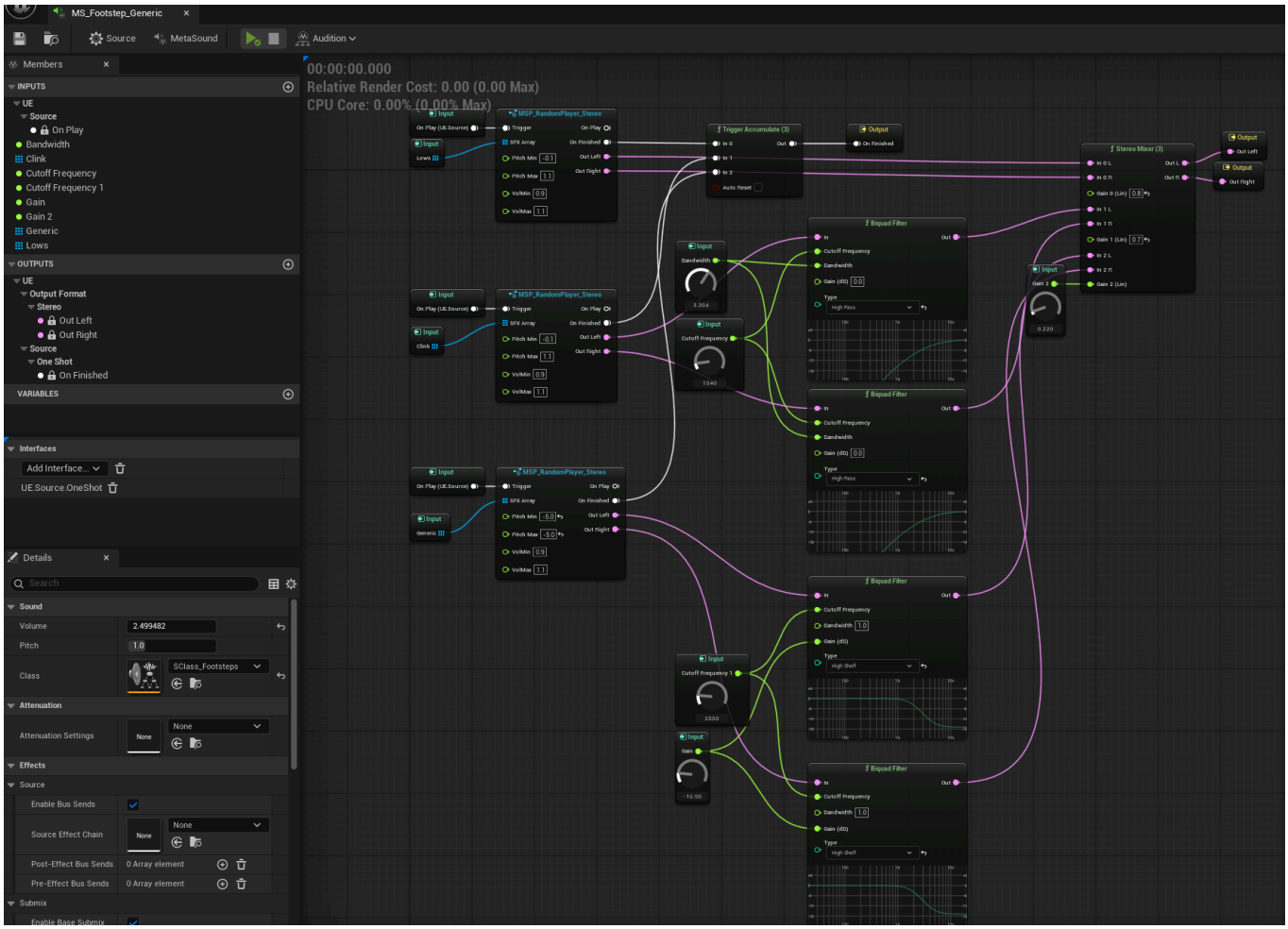
H.4 World Footstep Attenuation / Concurrency Assets





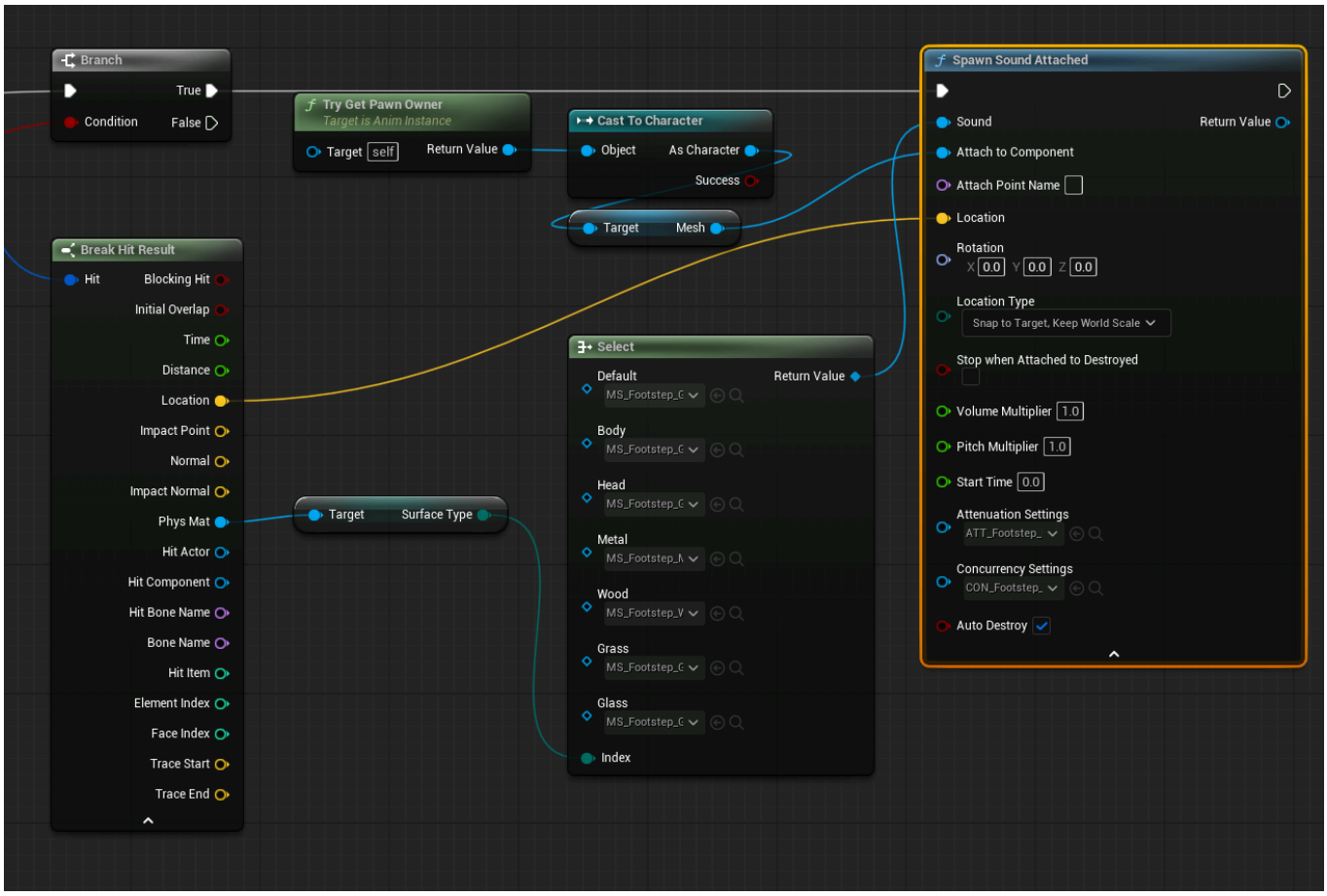
Caption:
World / third-person footstep attenuation and concurrency settings. Remote-player footsteps use a different playback treatment so they read correctly in the world without behaving like local first-person movement.

H.5 Reusable Random-Player Patch Settings



Caption:
Reusable random-player patch configuration used for footstep playback variation. This setup reduces obvious repetition while allowing the same array-selection pattern to be reused across multiple footstep assets.

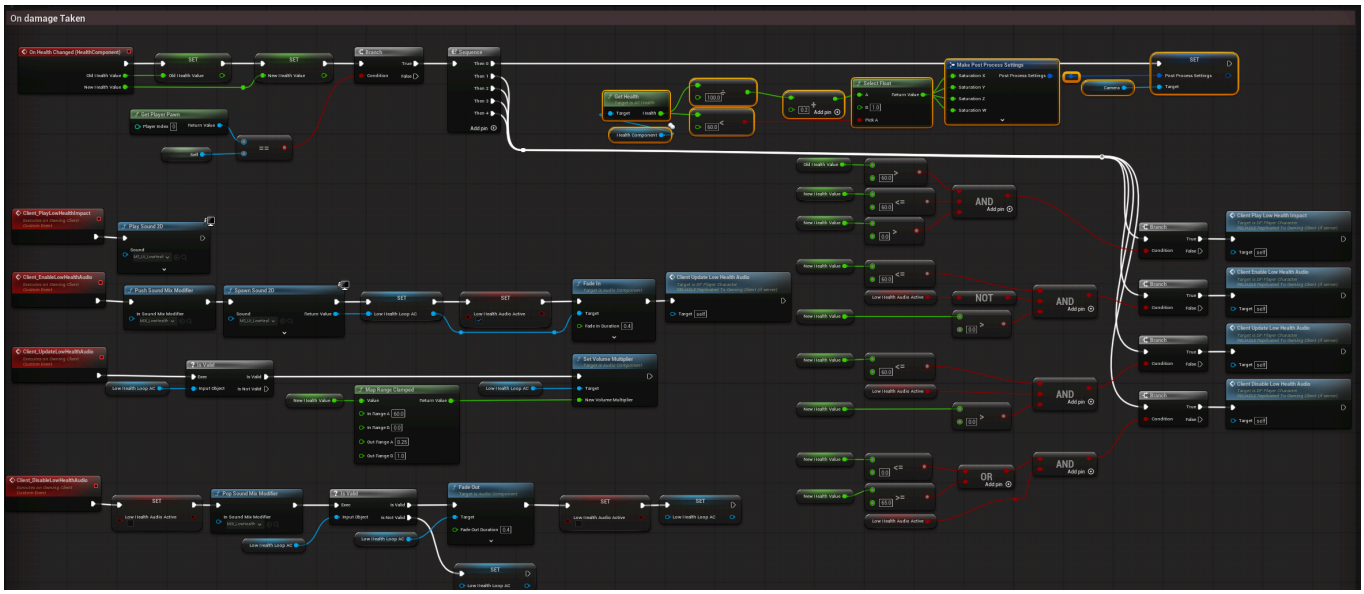
H.6 Footsteps Across Different Surfaces — Runtime Capture



Caption:
Runtime capture showing footsteps triggered across different surface types. This verifies that the material-routing system correctly switches between generic, wood, metal, and glass playback in-game.

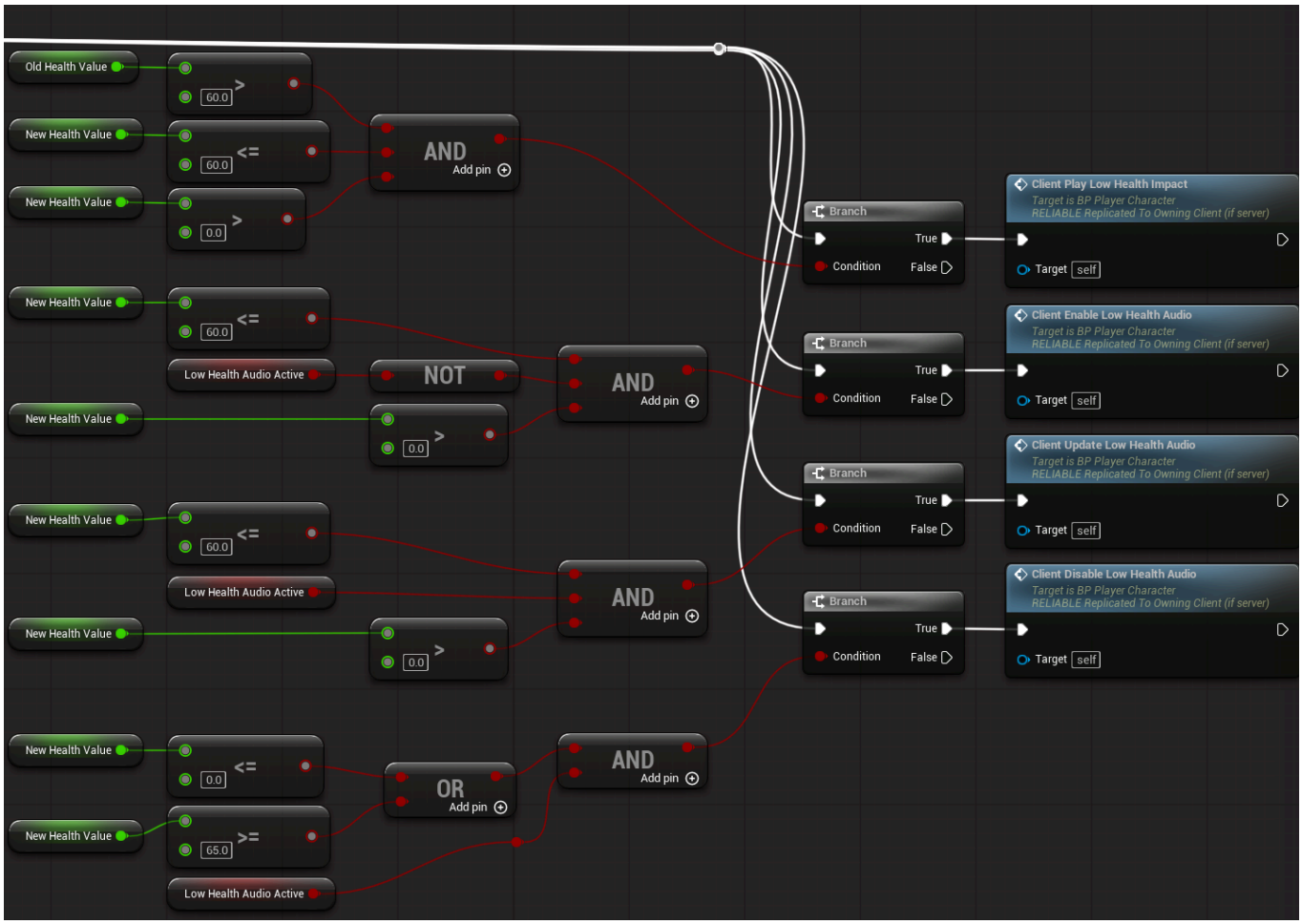
Appendix I — Player State Audio Evidence

I.1 BP_PlayerCharacter — On Health Changed Overview



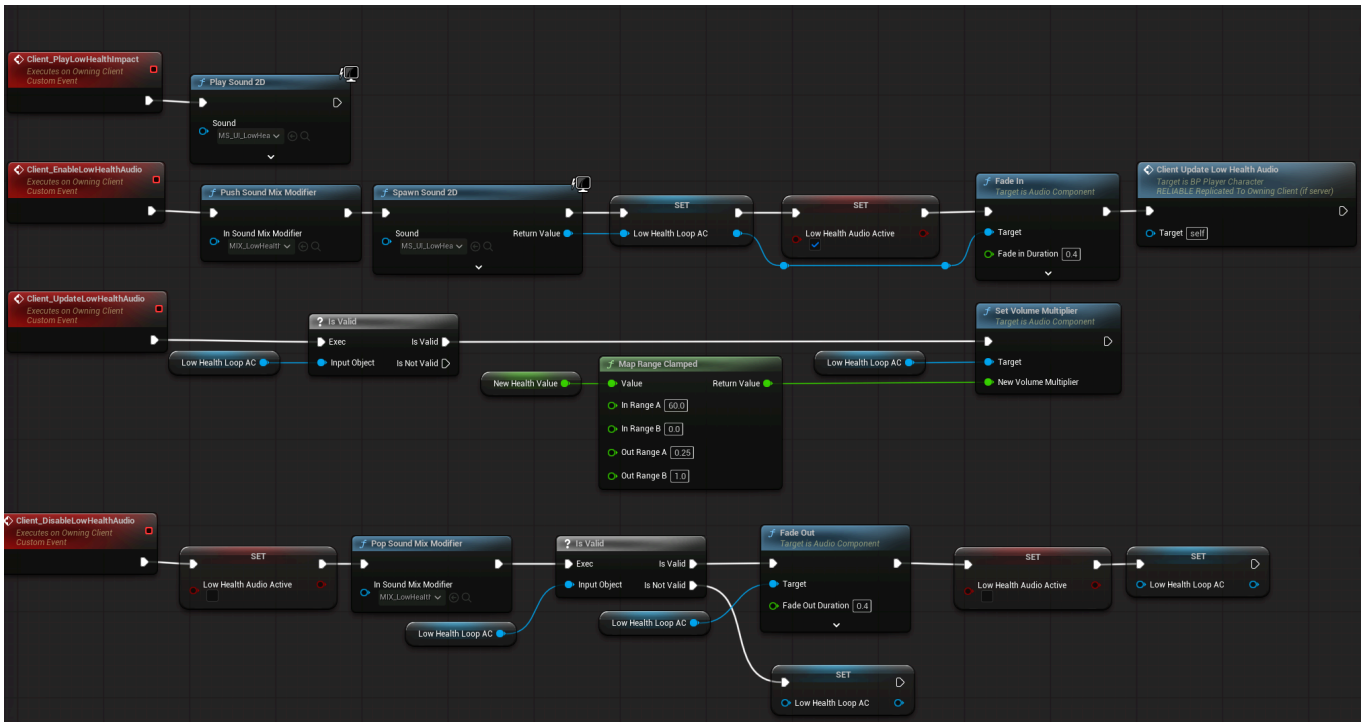
Caption:
Overview of the On Health Changed sequence in BP_PlayerCharacter. This graph handles local-player health-state audio, including low-health entry, persistent low-health mix activation, recovery clearing, and death-state cleanup.

I.2 BP_PlayerCharacter — Low-Health Threshold / Entry Sting Logic



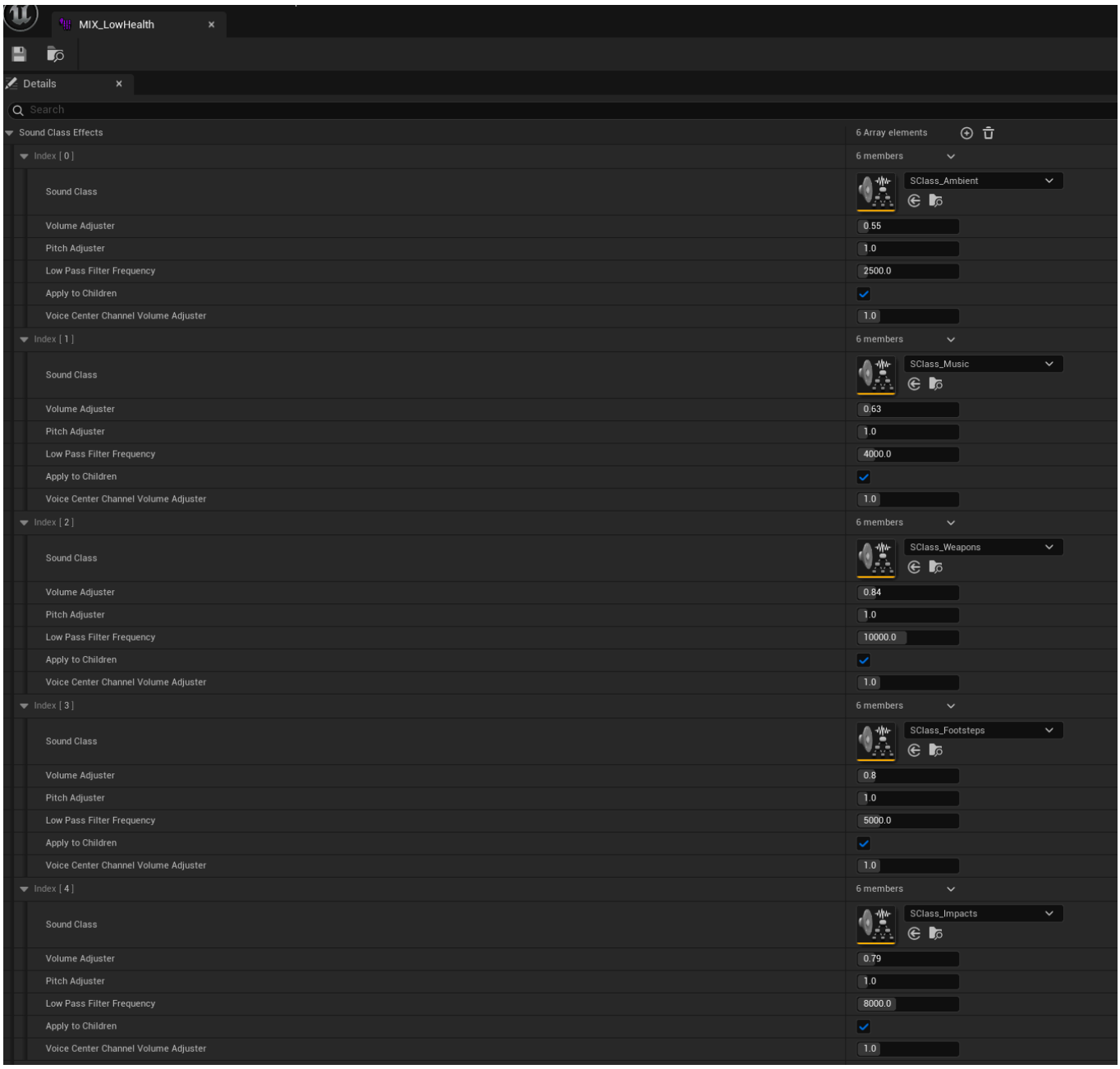
Caption:
Low-health threshold logic. The entry impact is only triggered when health crosses into the critical range, preventing the low-health one-shot from retriggering continuously while the player remains injured.

I.3 Client Low-Health Audio Events — Impact, Enable, Disable



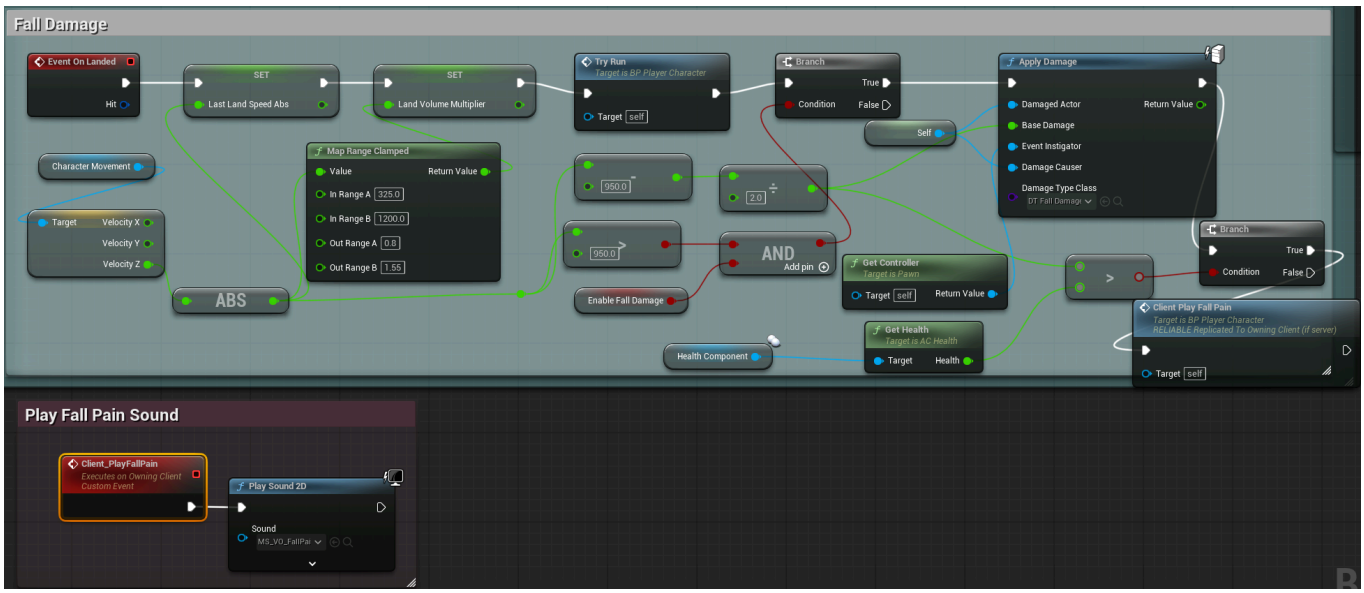
Caption:
Owning-client low-health audio events used to keep critical-health feedback local to the affected player. ClientPlayLowHealthImpact plays the short low-health impact cue, ClientEnableLowHealthAudio activates MIX_LowHealth while the player remains in critical health, and ClientDisableLowHealthAudio clears the mix after recovery or death so the game returns to normal audio presentation

I.4 MIX_LowHealth — Class Entries



Caption:
MIX_LowHealth class entries used to shape the low-health state. The mix reduces and filters selected gameplay categories while keeping critical feedback such as UI and core player information readable.

I.5 BP_PlayerCharacter — Damaging-Fall Branch



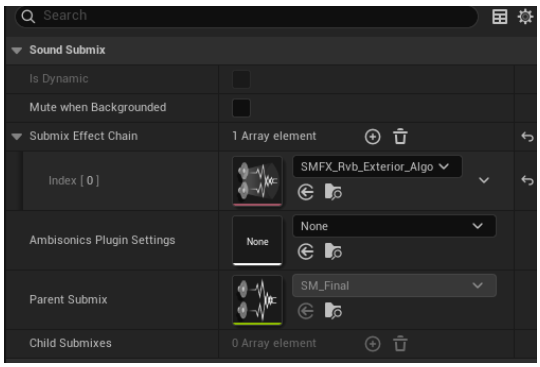
Caption:
Damaging-fall branch in BP_PlayerCharacter. Fall pain is triggered from the actual damaging-fall logic rather than from a data-table-only path, so the sound only plays when the landing qualifies as a painful fall event.

I.6 Death-State Cue Trigger

Caption:
Death-state cue trigger. Death audio is treated as a separate event from low-health audio, allowing the low-health mix to clear and the death cue to take priority when the player dies.

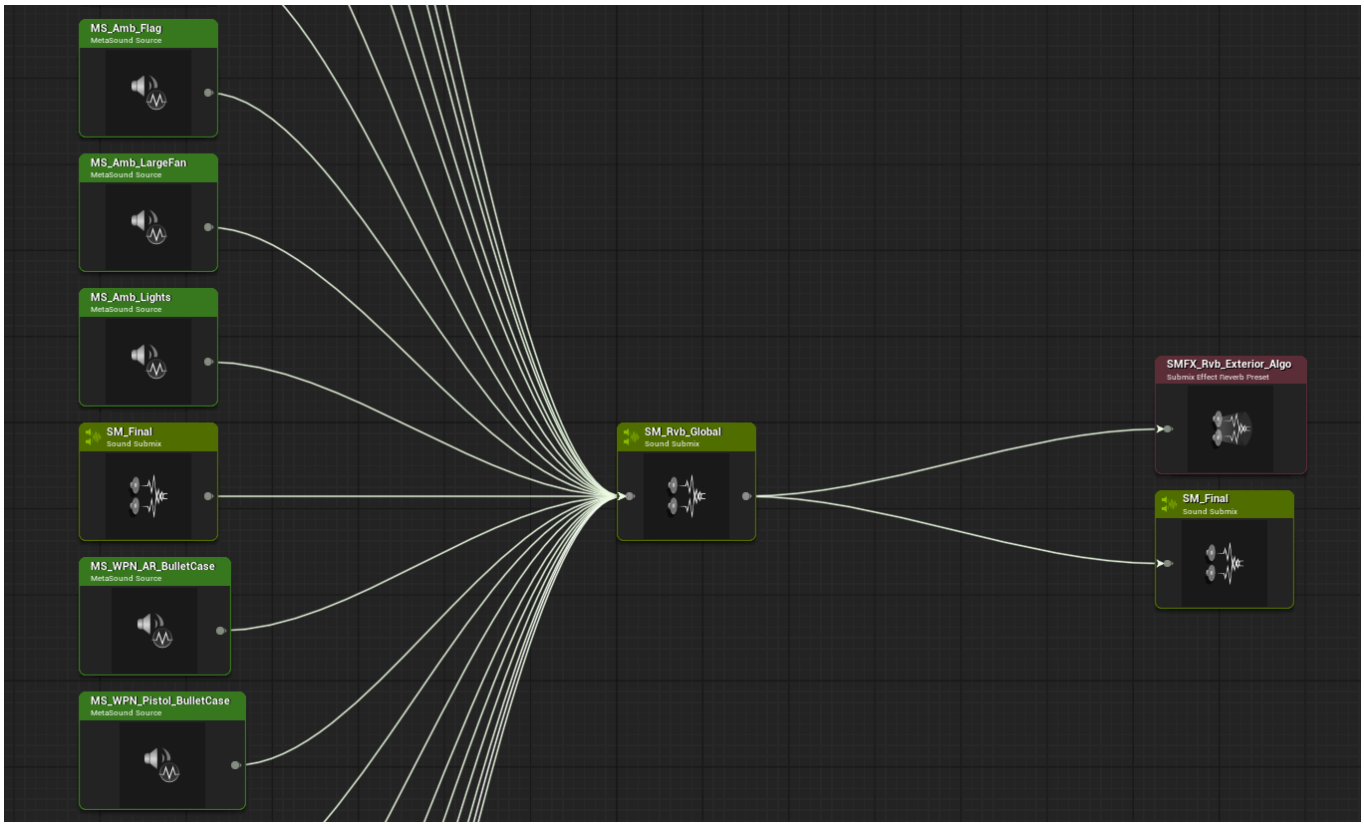
Appendix J — Reverb and Interior Override Evidence

J.1 SM_Rvb_Global — Effect Chain Overview



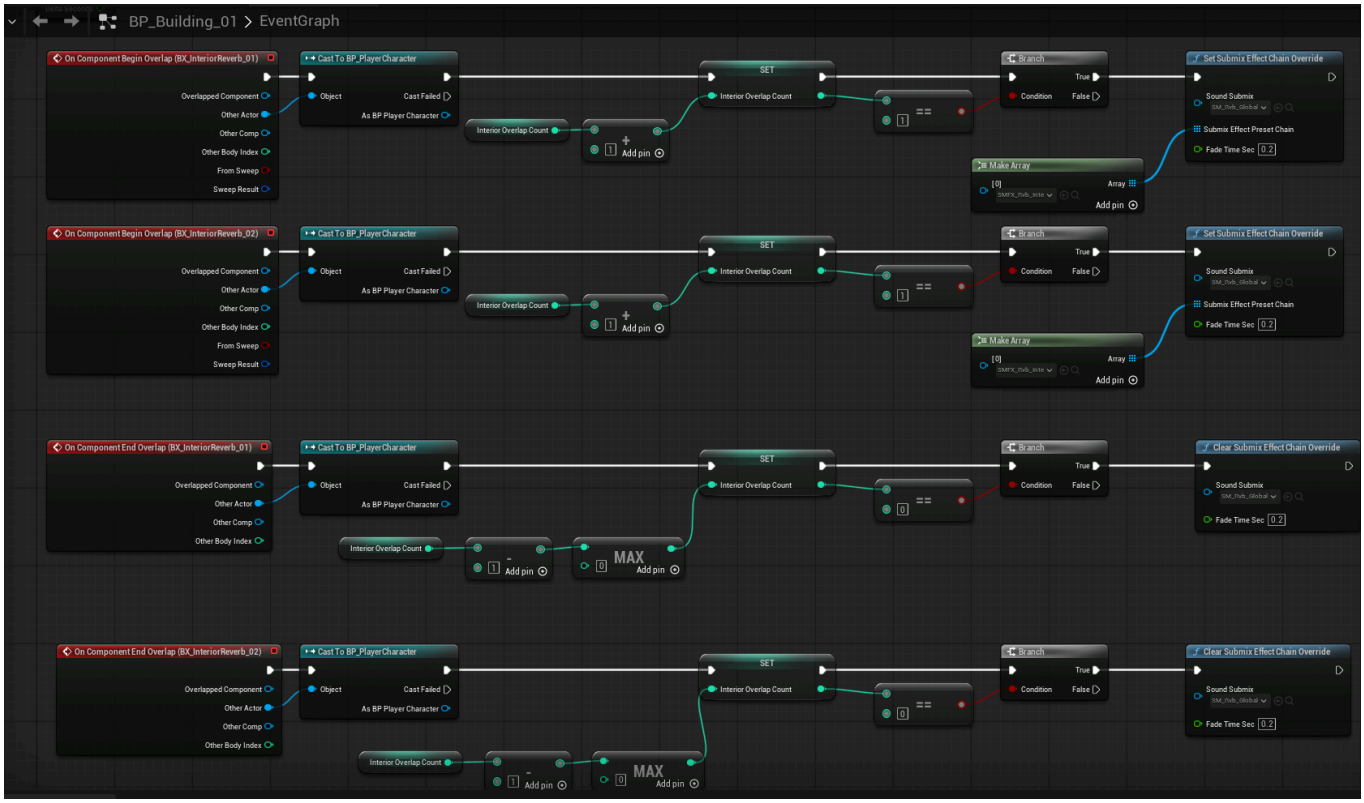
Caption:
Overview of SM_Rvb_Global, the shared project reverb return path. This submix carries the default exterior reverb chain and acts as the central destination for sounds that need environmental reverb treatment.

J.2 Representative MetaSound Send into SM_Rvb_Global



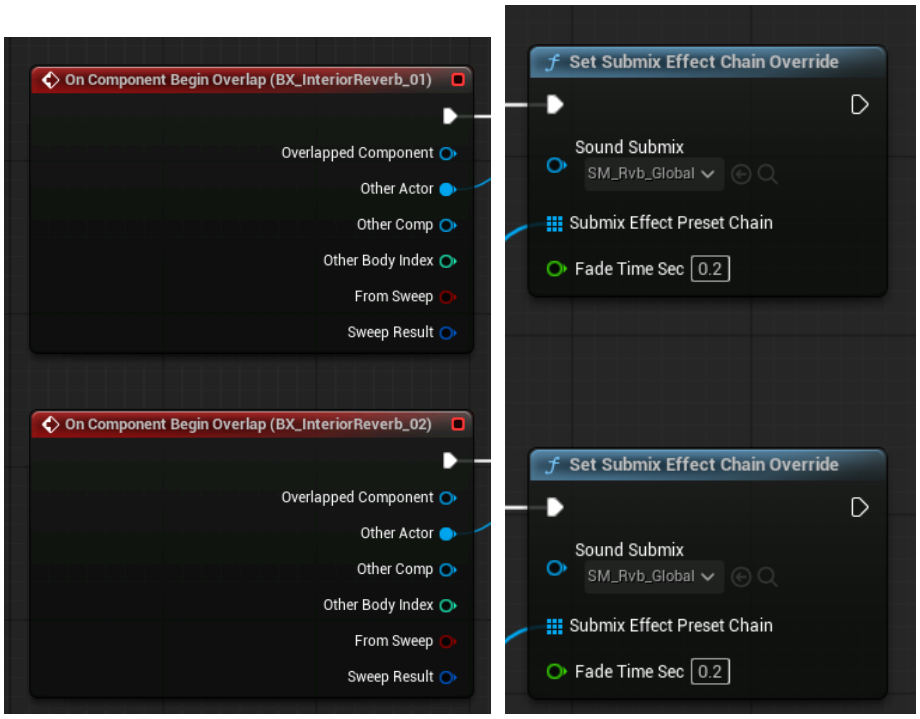
Caption:
Representative MetaSound send routing into SM_Rvb_Global. Sounds that need environmental reverb contribution are sent into the shared reverb path, allowing the project to control reverb behaviour centrally instead of building separate bespoke reverb logic into every asset.

J.3 BP_Building — Interior Overlap Graph Overview



Caption:
Overview of the BP_Building interior reverb overlap logic. Interior box overlaps are used to control when the project switches from the default exterior reverb chain to the interior reverb chain.

J.4 Set Submix Effect Chain Override

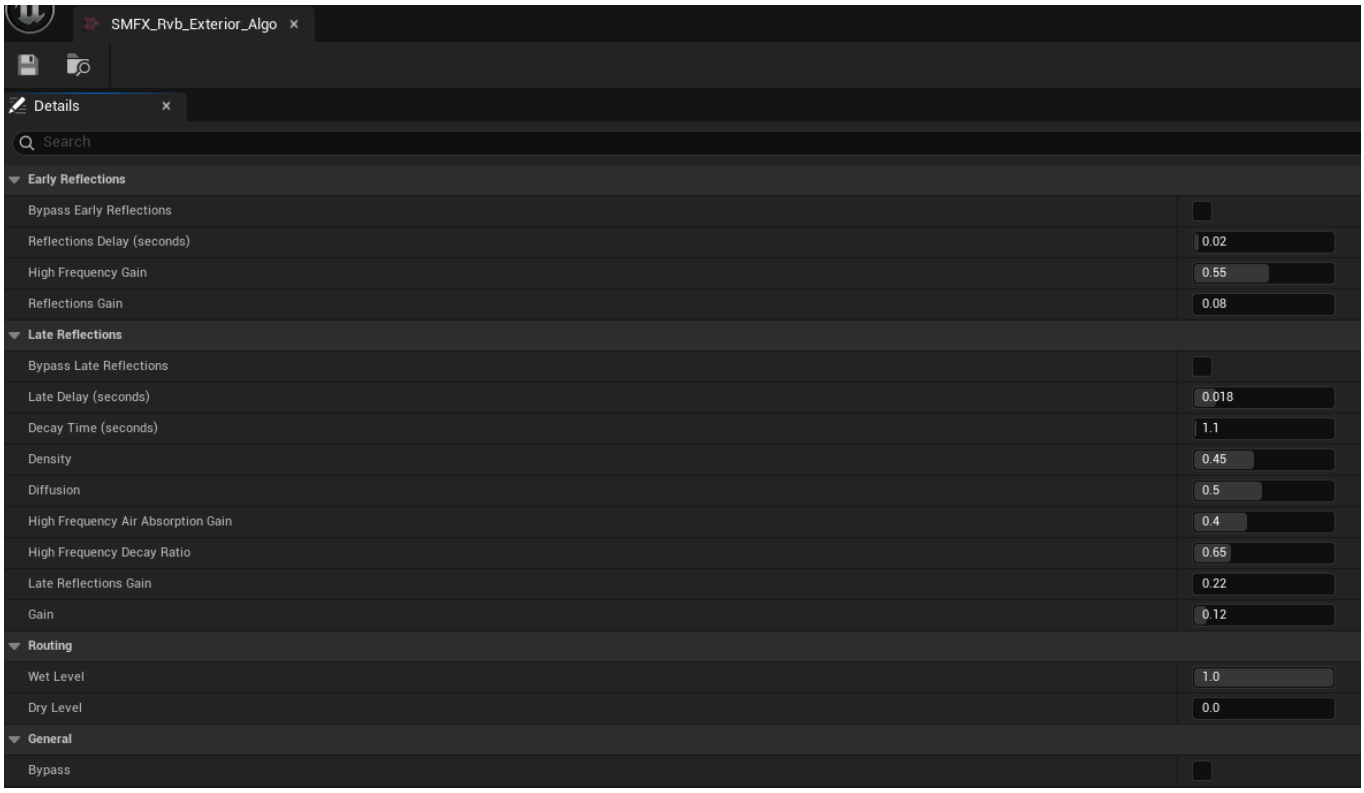


Caption:
Interior-entry logic using Set Submix Effect Chain Override. When the player enters an interior region, the shared reverb submix is overridden with the interior reverb effect chain, allowing indoor spaces to change the acoustic response without changing every individual sound asset.

J.5 Clear Submix Effect Chain Override

Caption:
Interior-exit logic using Clear Submix Effect Chain Override. When InteriorOverlapCount returns to zero, the override is cleared and SM_Rvb_Global returns to the default exterior reverb chain.

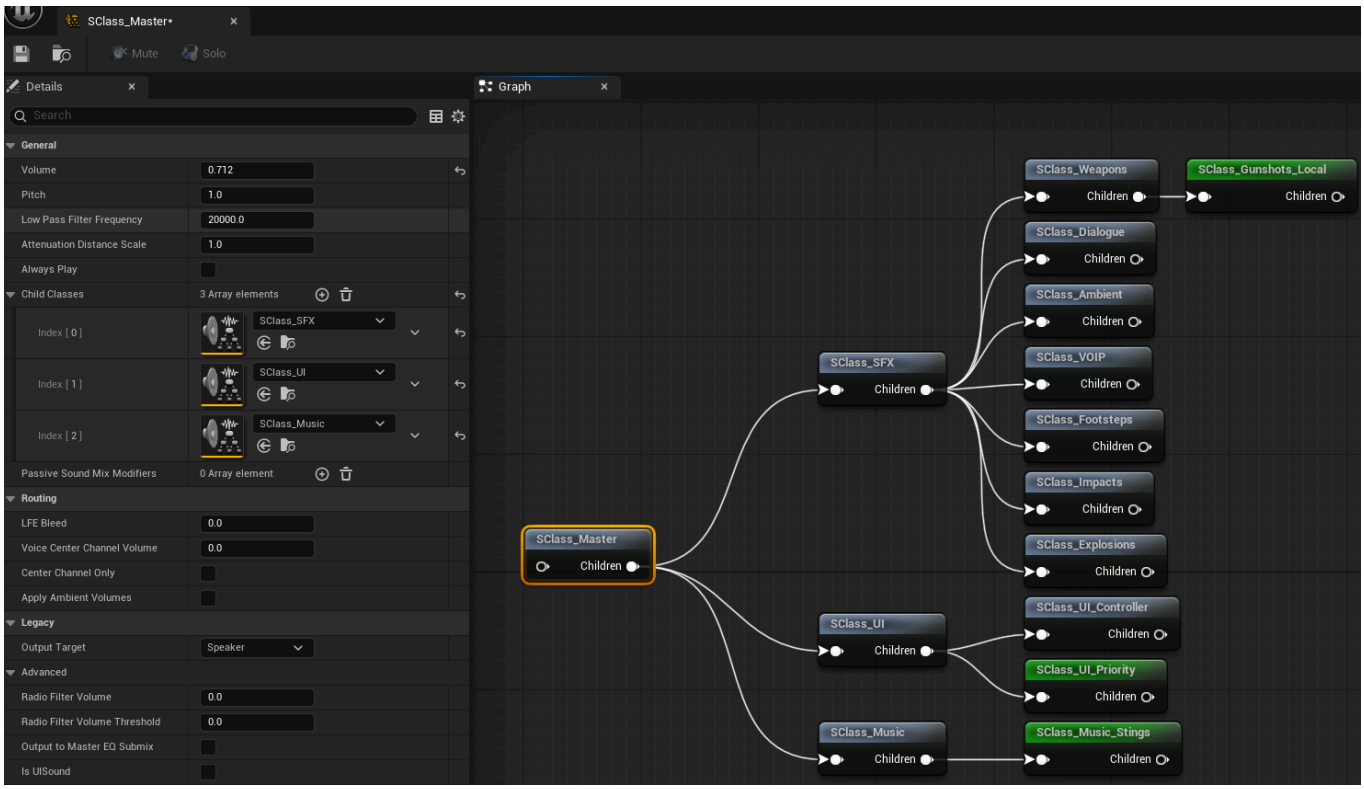
J.6 Exterior vs Interior Reverb Preset Comparison



Caption:
Exterior and interior reverb presets shown side by side. The project uses one shared reverb send path, then swaps the submix effect chain depending on whether the player is outside or inside.

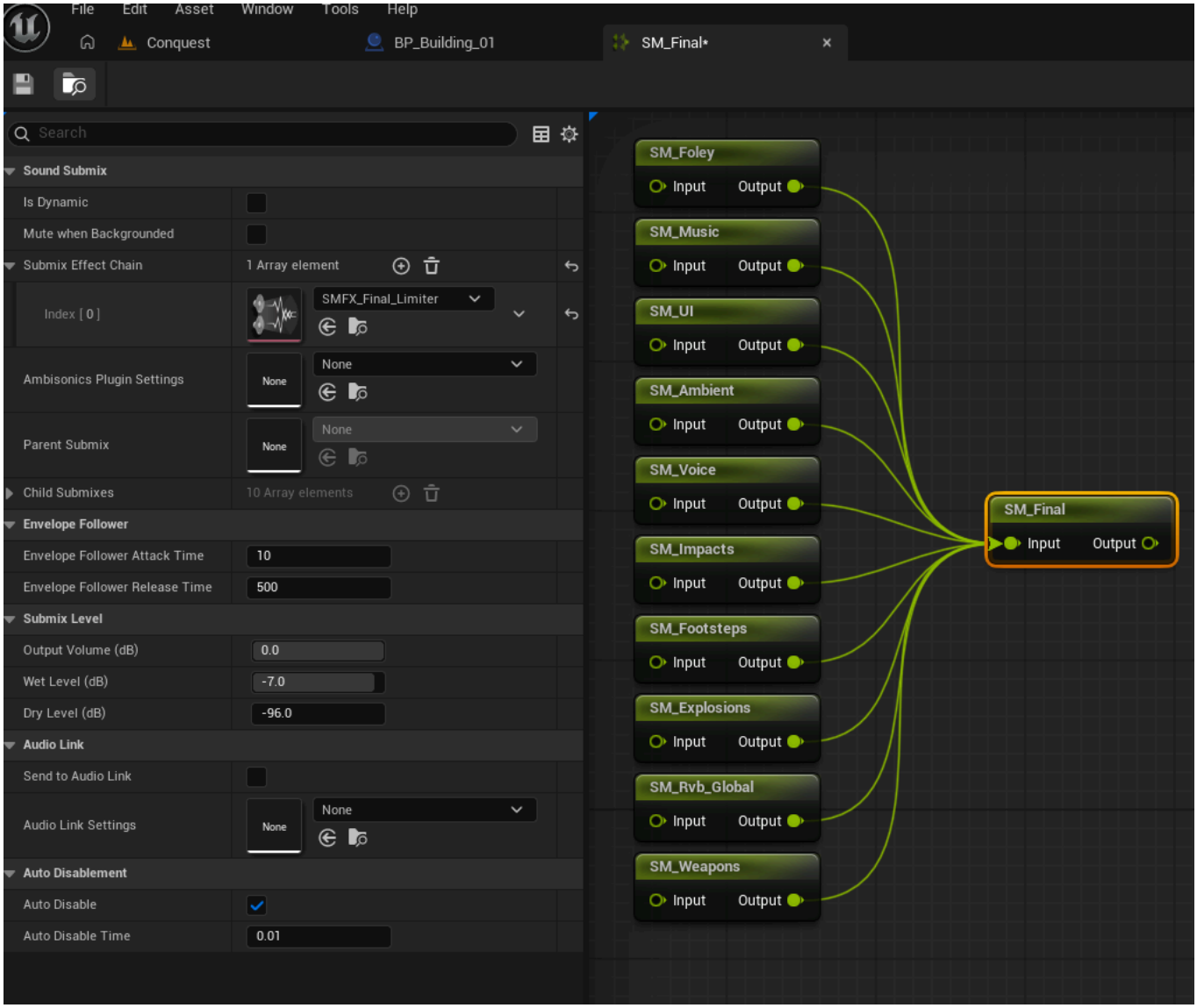
Appendix K — Routing, Dynamic Mixing, and Output Chain Evidence

K.1 SoundClass Graph — Main Routing Hierarchy



Caption:
SoundClass hierarchy showing the main project audio categories. The structure separates weapons, footsteps, impacts, explosions, ambience, UI, music, and related child classes so the mix can be controlled at category level rather than through isolated asset gain only.

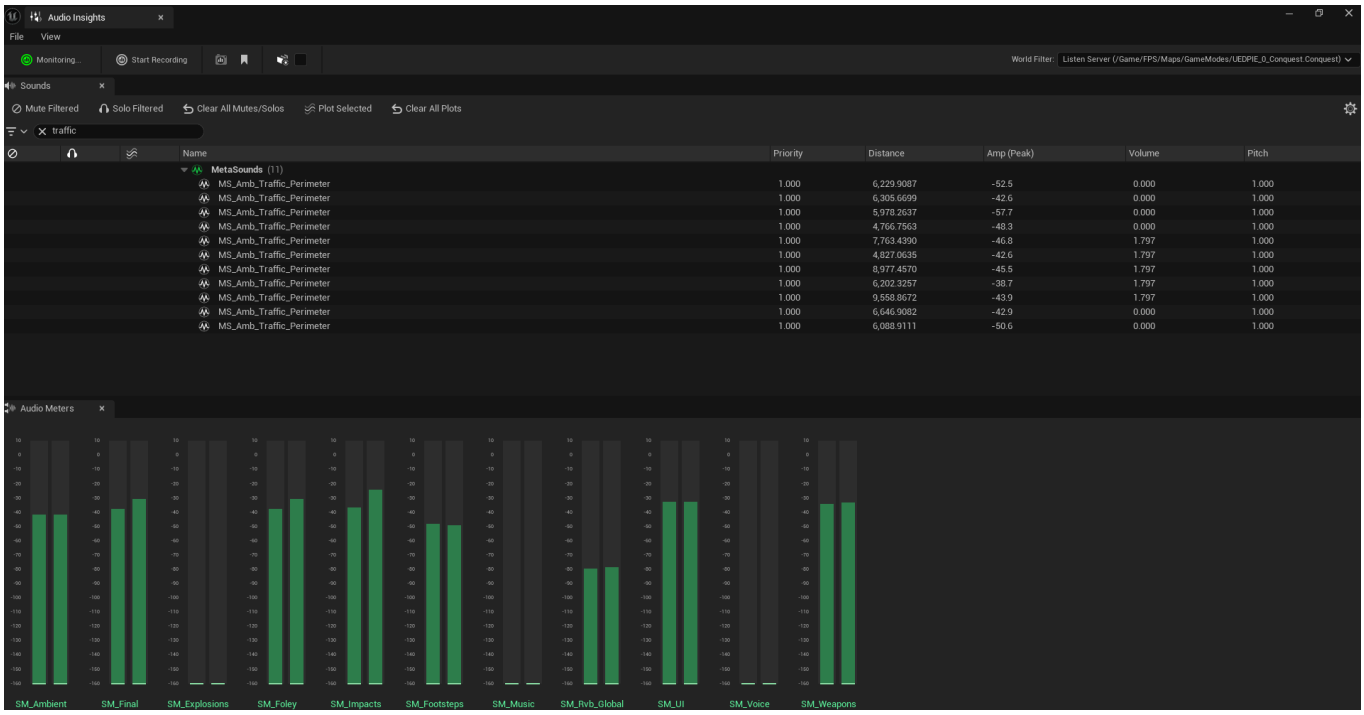
K.2 SM_Final — Submix Graph Overview



Caption:

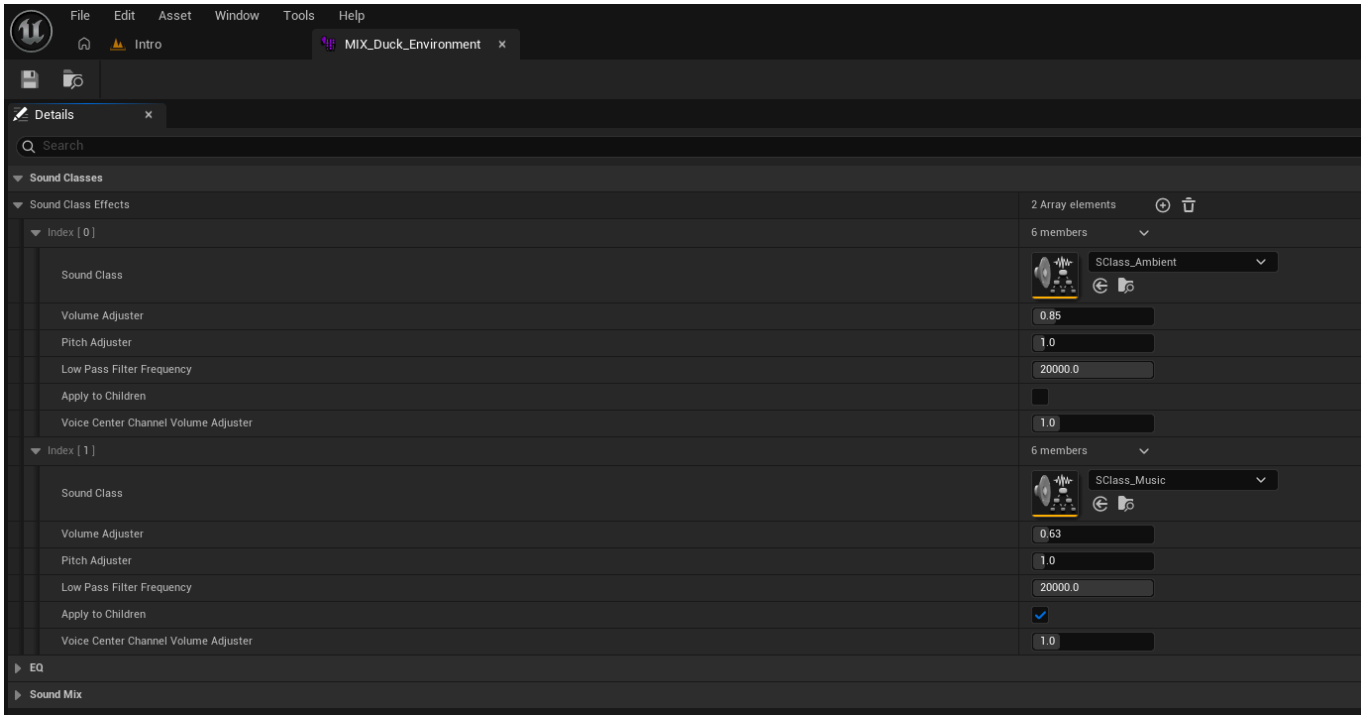
SM_Final submix graph showing the main project stems feeding into the final output path. This structure allows weapons, footsteps, impacts, explosions, ambience, UI, music, and reverb to be monitored and balanced as separate stems before final output.

K.3 Audio Insights — Active Stem Meters During Gameplay



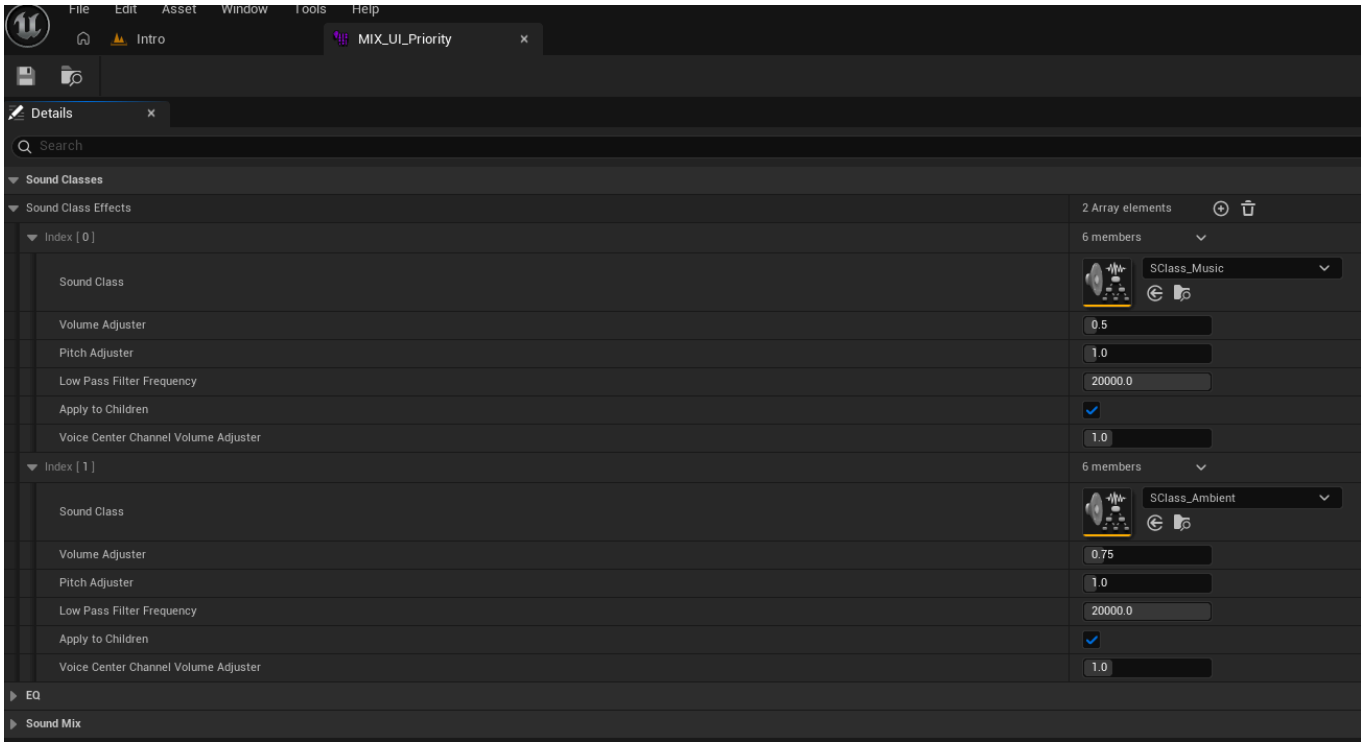
Caption:
Audio Insights Audio Meters showing active stem behaviour during gameplay. This verifies that the routing structure is functioning at runtime and that key categories can be monitored separately during combat, movement, UI, ambience, and reverb playback.

K.4 MIX_Duck_Environment — Local Gunshot Ducking



Caption:
Combat-focus mix used by local gunshot playback. MIX_Duck_Environment reduces background space during local firing so weapons remain readable without making every remote or world gunshot force the whole mix to pump.

K.5 MIX_UI_Priority — Priority Event Ducking



Caption:
Priority UI mix used for objective, countdown, or result events. This mix clears space for important gameplay information without applying the same behaviour to routine UI sounds such as menu highlights.