



CORE RX
ZERO-CODE XR FRAMEWORK

CORE RX

TECHNICAL DOCUMENTATION

English edition - source r147

CORE RX - FRAMEWORK

TECHNICAL DOCUMENTATION

English edition



Reference source: r147 (47 scripts) - Unity 6000.5.4f1
August 2026 - MAD-VF - Fabio Cecchini - www.mad-vf.store

Colophon and notices

This document accompanies the r147 source of CORE RX and is its complete map: every rule, every assembly contract, every silence and every constant, with a file:line reference. It does not contradict the user guide: it complements it. The guide remains the home of the usage and of the knobs (its chapter 16 is the catalog of the inspector fields, recalled here sheet by sheet); this documentation is the home of the behavior of the code. The tooltips of the fields live in chapter 16 of the user guide, quoted VERBATIM from the source; here every entry carries the exact identifier of the r147 inventory.

The rNNN references in the text (r104-r147) are the internal development revisions of the framework, test numbers on real hardware: they are not versions of the product. Updated distributions of the framework may shift some line numbers: the pair of file plus identifier of every entry remains the good reference. Assistance on the framework is included: email me at store@mad-vf.store from the email of your store account (the one of the purchase, so I recognize you right away) and I reply ASAP. Custom scenes, prefabs and custom modifications are on quote: same email with your need, and you receive a time and cost estimate ASAP. Work is done to a verifiable delivery. MAD-VF - Fabio Cecchini - www.mad-vf.store

CONTENTS

PART I - FOUNDATIONS	6
CHAPTER 1 - ARCHITECTURE AND SCENE FLOW	7
CHAPTER 2 - ASSEMBLY CONTRACTS	8
CHAPTER 3 - SOURCE AND DATA CONVENTIONS	9
PART II - SYSTEMS	10
CHAPTER 4 - BOOT, SCENES AND RETURNS	11
CHAPTER 5 - DISPLAY: THE FOUR MODES	12
CHAPTER 6 - PLAYER: LOCOMOTION, AVATAR, FINGERS, GAZE	13
CHAPTER 7 - VIVE FBT AND STEAMVR	14
CHAPTER 8 - PICO FBT	15
CHAPTER 9 - COMBAT AND MAGIC	16
CHAPTER 10 - WEAPONS AND GRAB	17
CHAPTER 11 - UI, HUD, AUDIO AND FEEDBACK	18
CHAPTER 12 - MIRRORS	19
CHAPTER 13 - DIRECTION, WAVES, DATA AND NETWORK	20
PART III - SOURCE REFERENCE	21
CHAPTER 14 - THE 47 FILE SHEETS	22
14.01 - RX_Abyss.cs	22
14.02 - RX_AudioEnding.cs	22
14.03 - RX_AvatarBuilder.cs	23
14.04 - RX_Blink.cs	26
14.05 - RX_BootSettings.cs	26
14.06 - RX_Character.cs	27
14.07 - RX_ComboManager.cs	28
14.08 - RX_DataBridge.cs	29
14.09 - RX_DisplayApplier.cs	30
14.10 - RX_EnemyAI.cs	31
14.11 - RX_FingerPoser.cs	35
14.12 - RX_GameDirector.cs	36
14.13 - RX_Haptics.cs	37
14.14 - RX_HitZone.cs	38
14.15 - RX_HowToPlay.cs	38
14.16 - RX_HowToPlayLauncher.cs	39
14.17 - RX_LaserGate.cs	39
14.18 - RX_LevelMusic.cs	40
14.19 - RX_MagicCaster.cs	40
14.20 - RX_MagneticGrab.cs	42
14.21 - RX_MirrorClone.cs	43
14.22 - RX_MirrorHandle.cs	44
14.23 - RX_Persistent.cs	44
14.24 - RX_PicoFBT.cs	44
14.25 - RX_PicoTrackerAdapter.cs	46
14.26 - RX_PlayerLocomotion.cs	47
14.27 - RX_RemoteLog.cs	49
14.28 - RX_SafeSpawner.cs	50
14.29 - RX_SecondWind.cs	51
14.30 - RX_Settings.cs	51
14.31 - RX_SimpleMirror.cs	52

14.32 - RX_UIGate.cs	52
14.33 - RX_UIManager.cs	52
14.34 - RX_UniversalProjectile.cs	53
14.35 - RX_VideoWipe.cs	54
14.36 - RX_ViveFBT.cs	54
14.37 - RX_ViveTrackerProfile.cs	57
14.38 - RX_WaveDirector.cs	57
14.39 - RX_WeaponDamage.cs	58
14.40 - RX_WeaponDropper.cs	60
14.41 - RX_WeaponRack.cs	60
14.42 - RX_WristHUD.cs	61
14.43 - MV_AlignRigToScene.cs	61
14.44 - MV_FrontOfCamera.cs	61
14.45 - MV_LoadSceneWithDelay.cs	61
14.46 - MV_SkipScene.cs	62
14.47 - MV_TrackerCube.cs	62
APPENDICES	63
APPENDIX A - THE 33 HOWTOPLAY PHRASES (ENGLISH AND ITALIAN)	64
APPENDIX B - THE TABLE OF SILENCES	67
APPENDIX C - THE CONSTANTS	68

CHAPTER 7

VIVE FBT AND STEAMVR

The engine is `RX_ViveFBT` (sheet 14.36): it carries the tracked ankles onto the IK targets of the leg rig with exponential smoothing, teleport guard, radial offset along the hip-ankle axis, kick boost, extension assist and cap, and the hammer foot on an orthonormal frame with anti-flip hysteresis. It is also the console of the locomotion: `fbtMoveThreshold`, `trackerFrozenTimeout` and `hookCenterMaxRadius` live here and are pushed onto the locomotion every frame (the locomotion is mounted at runtime and has no persistent inspector: the editable home is this one). COMPLETE PROCEDURE OF THE `IsTracked` ACTIONS (the debt left by the seal of the guide, settled here).

Step 1, the feature: enable RX Vive Tracker Profile among the OpenXR Standalone features (Interaction category). The feature lives on the `XR_HTCX_vive_tracker_interaction` extension: without the extension active in the runtime it is not created (`RX_ViveTrackerProfile.cs:56`), and the reference runtime is SteamVR.

Step 2, the layout: the feature registers the `RX_ViveTracker` device with a matcher on the `XRInput` interface and the VIVE Tracker product (`RX_ViveTrackerProfile.cs:65`). The layout exposes the `devicePose` control (usage Device, noisy) plus the standard controls of every OpenXR device, among them `isTracked` AT THE TOP LEVEL of the device.

Step 3, the roles: two logical devices, `left_foot` with the Left characteristic and `right_foot` with Right (`RX_ViveTrackerProfile.cs:87`): it is the characteristic that tells the sides apart in the binding masks.

Step 4, the pose actions: the tracker position actions of the locomotion (`leftTrackerPosAct` and `rightTrackerPosAct`) bind to `devicePose/position` of the device of their own side; presence is measured from the resolved controls (`RX_PlayerLocomotion.cs:138`).

Step 5, the `IsTracked` actions: two dedicated actions, one per side, bound to the TOP-LEVEL `isTracked` control of the device (NOT to the child inside the `devicePose` group: that child proved dead, `r105`), and wired into `tLeftIsTrackedAct` and `tRightIsTrackedAct` of `RX_ViveFBT` (`RX_ViveFBT.cs:26-27`).

Step 6, the behavior: both wired = certification authority: 2 out of 2 at value 1 and the hook is instant and certified, with no step ritual; any other value = no FBT, by design (the step fallback stays off). Only one wired = authority off for safety, and here too the step fallback comes back active (canon `r128`: one side only = warning and fallback). None wired = back to the step handshake: both trackers moved beyond threshold, far from each other, at a plausible height (`RX_ViveFBT.cs:201-222`, `RX_PlayerLocomotion.cs:167`).

Step 7, the unhook: always and in any case via surveillance: presence from the controls and the frozen-pose VITA (`trackerFrozenTimeout`, measured in the local space of the parent: the movement of the rig does not mask a dead tracker, `RX_PlayerLocomotion.cs:138-147`).

Step 8, the check in the field: `MV_TrackerCube` with its traffic light (green tracked, red not tracked, gray binding not resolved) and the `MV_CUBO` log (sheet 14.47). The dongle and tracker scenarios were field-tested on Vive AIO on August 2 (0/0/1) and re-verified on SteamVR at the platform-0 pass, as the tooltip of the field records (`RX_ViveFBT.cs:26`), quoted in full in chapter 16 of the guide.

CHAPTER 9

COMBAT AND MAGIC

The rules of the hit, the ones the player lives, stay in 6.5 of the user guide: that is their home, here is the engine that enforces them. The judge of every contact is `RX_WeaponDamage` (sheet 14.39), mounted on everything that wounds: the player's punches and kicks (manufactured by the builder), the enemies' hitboxes (manufactured by the AI), the weapons.

It measures the true speed of the gesture in unscaled time, demands the re-arm between one hit and the next (pull back and charge in again: no damage by rubbing; the fixed-damage kicks of the Second Wind do without it, lines 112-113 and 121), applies the per-target cooldown, the halving of the open-hand punch, the sides (a free weapon or a weapon in the player's hand never hits the player, lines 128-129; the weapon of the armed enemy wounds the player instead: lines 119-154), the melee window of the enemy (one attack, one hit) and the wear of the weapons with finite uses. The body zones are `RX_HitZone` markers (sheet 14.14).

The player mounts twelve (lines 193-205 of the builder): damage on the head, on the trunk and on the thighs; parry on forearms and shins, which stop melee and magic; and hands and feet, which stop magic only (lines 202-205). The ground enemy mounts eight (`RX_EnemyAI.cs:96-104`): damage on head, trunk and thighs and parry on forearms and shins, without parrying hands and feet. The flyer mounts none (branch of line 57): it takes damage on the whole body. A body with `usesHitZones` takes nothing outside a zone: you hit where the body declares it can be hit.

The opposing brain is `RX_EnemyAI` (sheet 14.10): two natures (ground and sky), distance bands (melee within `attackRange`, magic in the `castMin-castMax` band, pursuit beyond), attack tokens from the direction, retreats at 50 percent after the hit, kamikaze with the telegraph given by the minimum dive distance (the armed flyer ignores it: the dive is its only move, line 248). Magic flies on `RX_UniversalProjectile` (sheet 14.34), the same for everyone: homing with a dodge window, magic-against-magic cancellation, speed-based damage for the player's magic and the point-blank BLAST at double damage with all the rewards.

The combos are Knuth-Morris-Pratt automata on `RX_ComboManager` (sheet 14.07): sequences in the inspector, generic moves that subsume the lateral ones, rewards in damage, points, mana and heal. The Second Wind (sheet 14.29) is the last word: slow motion in real time, one landed hit to get back up, one single time per match. The player's magic leaves from the hand: `RX_MagicCaster` (sheet 14.19) governs the mana with the scene policies (full in the menus, zero in the ending), demands a free hand and no UI under aim, spends in advance and refunds only in the right cases (pause, punch within the grace), and on release hurls the magic along the axis of the arm: you aim with the arm, which the controller leads.

The canon is sealed: free laser and free hand.

CHAPTER 13

DIRECTION, WAVES, DATA AND NETWORK

The direction of the match is twofold. `RX_GameDirector` (sheet 14.12) governs the state transitions: start of the arena, pause with a coherent freeze (`timeScale` at zero with the previous value returned exactly: the states in slow motion resume from where they were), ending with full restore, and exit from the game. `RX_WaveDirector` (sheet 14.38) governs the rhythm: match time, spawn lanes below the maximums every 5 seconds, attack tokens in rounds with a Fisher-Yates shuffle, immediate recalculation at every death and a short retry when everyone beats a retreat.

The difficulty factor is one single formula (`DifficultyFactor`: `difficulty` and `arenaLevel`, minimum 0.1): it multiplies the health of the enemies and the damage taken by the player (`RX_Character.cs`:75 and 121); the heals of combo and blast follow the difficulty by another road, in fixed units on the registry. The obstetrician is `RX_SafeSpawner` (sheet 14.28): ground pose sampled on the `NavMesh` far from the player, air pose on the ring toward the edge, and three choreographed entrances (portal, emergence, rise) with the enemy frozen. The undertaker is `RX_Abyss` (sheet 14.01): rescue for the player as long as the health is above the minimum health threshold (at the threshold or below, end of match), incinerator for everything else.

The data are three JSON archives parsed by `RX_DataBridge` (sheet 14.08): bare arrays, wrapper at runtime, last entry with equal IDs, global fallbacks for the enemies' clips and phrases. The reference files live in `JSON_build` (players, weapons, enemies) and respect the shapes of the three profiles; they also carry keys prepared for future editions (the `ai_pattern` block of the enemies, weakness, `staggerImpact` and the like) that this engine does not read: filling them in changes nothing (note of honesty of the guide, appendix E).

The network is deliberately minimal: `RX_RemoteLog` (sheet 14.27) captures the logs and ships them as a raw-text POST to a PHP endpoint; the `Server_debug/log.php` script accepts the body up to 5 MB and writes it into `log.txt` next to itself, answering ok (`log.php`:2-7). No accounts, no telemetry: a debug channel on command, and the quit is guaranteed even with a dead server.

CHAPTER 14

THE 47 FILE SHEETS

(for every file: role, the count of the fields with a cross-reference to chapter 16 of the guide and, before any other entry, the hidden fields if any; then RULES, CONTRACTS, PLATFORM CONSTRAINTS, SILENCES and CONSTANTS in one single thread, in the order of the source, every entry with its file:line reference. The unit of count of the census is the entry, 461 for the fields: 431 entries among the inspector fields, which are 433 fields with two paired couples at 14.26; and 30 entries among the hidden fields, which are 40 fields with ten paired couples, five at 14.24 and five at 14.36, always printed with both names)

14.01 - SHEET RX_Abyss.cs

Role: fall sensor of the arena. It is a trigger volume to be mounted under the floor: everything that enters it is sorted by nature (player, enemy, corpse, magic, weapon) and treated accordingly. For the player it is a rescue service, for everything else it is an incinerator.

Fields: 4 inspector fields, all treated in chapter 16 of the user guide (cross-reference). No hidden fields.

CONTRACT (RX_Abyss.cs:13) **RX_Character on the parent:** the recognition uses a GetComponentInParent lookup for RX_Character: player and enemies must expose the component in the hierarchy above the collider that touches the trigger.

RULE (RX_Abyss.cs:16) **player fall debounce:** a fall within 1 second of the previous one is ignored with a silent return; the 1f constant is wired into the source.

RULE (RX_Abyss.cs:18) **rescue gate:** with CurrentHp above rescueMinHpUnits, penalty (ApplyFallPenalty) and teleport kick in; otherwise the match ends directly. Second Wind is never triggered by the abyss.

CONTRACT (RX_Abyss.cs:20) **RX_PlayerLocomotion.TeleportRig:** the teleport is delegated to the locomotion found with FindFirstObjectByType; destination playerRespawn, or the origin of the world if the field is empty.

SILENCE (RX_Abyss.cs:22) **locomotion absent:** without RX_PlayerLocomotion in the scene the penalty is applied but the teleport does not happen, with no signal.

CONTRACT (RX_Abyss.cs:24) **RX_GameDirector.GoToEndingScene:** the end of the match from the abyss goes through the GameDirector, if present in the scene.

RULE (RX_Abyss.cs:28) **corpses:** an already dead enemy that falls is only destroyed: no points, no wave counts.

RULE (RX_Abyss.cs:31) **flying enemies:** the groundFallBonus bonus rewards only the fall of non-flying enemies; a fallen flyer is worth no points.

CONTRACT (RX_Abyss.cs:31) **RX_WaveDirector.Instance:** score and removal from the wave count (RemoveEnemy with the flying flag) go through the WaveDirector, if present.

RULE (RX_Abyss.cs:33) **object cleanup:** what enters with the Magic or Weapon tag is destroyed: the abyss is also the bin of the lost weapons.

14.02 - SHEET RX_AudioEnding.cs

Role: director of the audio-only ending. It brings the camera back to the skybox, plays the optional intro phrase and then the ending clip, and returns to the Boot in any case: at the end of the clip, on a key press or in the absence of the source. Every wait runs in real time, immune to the time scale.

APPENDIX B

THE TABLE OF SILENCES

Sixteen points where the framework fails without saying anything: it is a choice (the frame is never sacrificed to a log), but it must be known. For every silence: where, what keeps quiet, how to notice it. The full detail lives in the sheet of the file.

1. RX_Abyss.cs:22, locomotion absent: the fall penalty is applied but the teleport does not happen. Symptom: the wounded player stays in the void.
2. RX_AudioEnding.cs:64, camera absent: the repositioning of the LOADING canvas is skipped, with no signal. Symptom: none visible in the audio ending, where the canvas lives switched on only inside the synchronous block (lines 41-45) and does not reach the screen; the sign standing still in the void is seen instead in the video twin, which keeps the canvas on during the preparation (RX_VideoWipe.cs:50-74).
3. RX_AvatarBuilder.cs:98, empty catch: every exception of the automatic tracker mounting is swallowed. Symptom: tracker references missing without errors in the console.
4. RX_AvatarBuilder.cs:123, arm bones missing: the arm rig is not born. Symptom: arms of the avatar that do not follow the hands.
5. RX_Blink.cs:32, camera absent: no overlay and no beats. Symptom: awakening without an eyelid.
6. RX_DisplayApplier.cs:72, arenaFloor empty: no boundary ring. Symptom: XR or AR without a ring on the perimeter.
7. RX_DisplayApplier.cs:78, bounds not found: without a Renderer or a Collider on the floor, no ring. Symptom: as above.
8. RX_DisplayApplier.cs:86, non-positive radius: margin wider than the floor, no ring. Symptom: as above, with small floors.
9. RX_EnemyAI.cs:66, non-humanoid ground enemy: no automatic hitboxes, zones and FirePoint. Symptom: enemy that neither strikes nor parries.
10. RX_Haptics.cs:40, empty catch: every haptic exception swallowed. Symptom: no vibrations, no error.
11. RX_HitZone.cs:12, null bone: Build returns null. Symptom: zone missing on a rig without the required bone.
12. RX_HowToPlay.cs:85, null operation: the return to the Boot does not happen (scene outside the Build Settings). Symptom: tutorial that does not exit.
13. RX_MirrorClone.cs:17, dependencies absent: no clone at the mirror. Symptom: mirror that frames the whole scene, real avatar included: the culling mask stays at the defaults, the narrowing of line 60 never arrives.
14. RX_SafeSpawner.cs:19, empty pool: no spawn from the lane. Symptom: arena without ground or air enemies.
15. RX_SafeSpawner.cs:35, attempts exhausted: 15 poses refused, no spawn. Symptom: rare spawns with a narrow NavMesh or the player at the center.
16. RX_SimpleMirror.cs:15, renderer absent: the mirror is not built. Symptom: inert mirror screen.

END OF PREVIEW

This free preview contains 11 pages taken exactly as they are from the official document: the cover, the title page, the colophon with the contacts up front, the complete contents, three system chapters (VIVE FBT and SteamVR; Combat and magic; Direction, waves, data and network), the first page of the 47 file sheets and the table of silences, plus the back cover here at the close.

THE COMPLETE TECHNICAL DOCUMENTATION

CORE RX - Framework - Technical Documentation, English edition: **72 A4 pages**. The engineering companion to the User Guide: thirteen chapters on the architecture and the systems, then the 47 file sheets - one per source script, with its role, its fields and every rule, contract, platform constraint, silence and constant - and three appendices. **1202 inventory entries**, each with its exact file:line reference to the r147 source: 461 knobs, 558 rules, 146 contracts, 12 platform constraints, 16 silences, 9 constants. What the code does, where it does it, and what it deliberately keeps quiet about: verified statement by statement against the source, page by page against the sealed Italian edition.

IN THE BOX

- The complete user guide in English and in Italian (84 pages each)
- The Technical Documentation in English and in Italian (72 pages each)
- The EULA in English and in Italian
- The Quick Reference in English and in Italian (4 pages, with the PLATFORM SWITCH checklist)
- The framework source: 42 scripts + 5 tools, with the 1202-entry source inventory

LICENSE AND ASSISTANCE

Framework updates are INCLUDED in the license (EULA, sec. 6) and distributed at no extra cost. Assistance on the framework is included: write from the email of your store account and I reply ASAP. Custom scenes, prefabs and custom modifications are on quote: you receive a time and cost estimate ASAP, and work is done to a verifiable delivery.

TRY IT FOR FREE BEFORE YOU BUY

FFRU DEMO RC is created with CORE RX as its engine: download it free from www.fightforresurrection.app/crx and play, to see first hand what you can create.

mad-vf.store

MAD-VF - Fabio Cecchini

Meta Quest - PICO - VIVE - SteamVR



CORE RX

TECHNICAL DOCUMENTATION

The complete map of what the framework does under the hood.

The r147 source is the source of truth: 47 scripts read line by line; every rule, every assembly contract, every silence documented with a file:line reference.

The four modes - VR, MR, XR, AR - explained by the code that governs them.

The full-body engines: PICO, VIVE, SteamVR - and Meta Quest via Link (platform O, VIVE trackers).

461 knobs, 558 rules, 146 contracts, 12 platform constraints, 16 silences, 9 constants: counted, one by one.

No black box: what you read here is what runs.

Assistance included: email me at store@mad-vf.store from the email of your account (the one of the purchase) - I reply ASAP. Customizations on quote: same email, a time and cost estimate

ASAP, verifiable delivery.

MAD-VF - www.mad-vf.store