AVCS Vehicle Conversion Guide

Tracked and Wheeled Customer Conversions | Updated 2 August 2026

This guide explains how to build a customer-owned vehicle on the AVCS conversion jigs without copying protected vehicle signatures or damaging the shared runtime contract. It is written for Roblox Studio developers who are comfortable importing meshes, positioning parts, editing attributes, and testing server/client behavior.

1. What the conversion jigs are

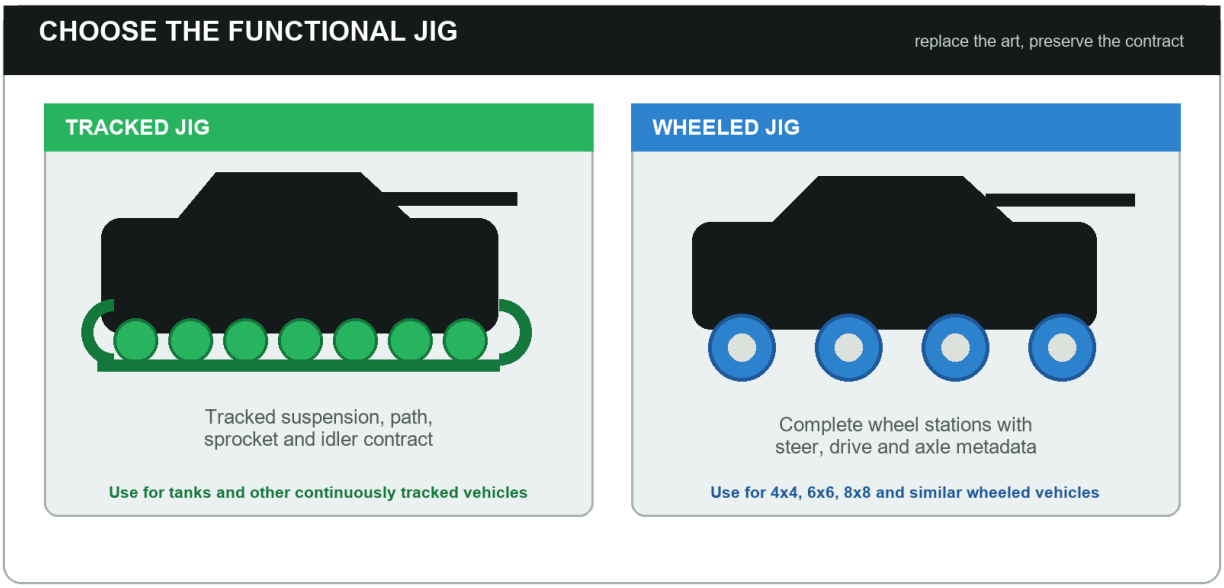

Figure 1. Choose the tracked or wheeled functional skeleton, then replace only the presentation geometry.

The package contains two intentionally unsigned templates in `ServerStorage.AVCSConversionTemplates`:

- `AVCS Tracked Conversion Jig`
- `AVCS Wheeled Conversion Jig`

Each jig contains replaceable primitive visual geometry plus the functional structure required by AVCS: hull and weapon pivots, seats, controls, collision volumes, damage zones, X-ray references, lights, recovery equipment, network objects, scripts, and per-vehicle specifications.

The generic runtime track-link asset in `Shared Reference Assets` is intentionally available for tracked conversions.

The jigs are not blank models. They are working contracts. Replace and reposition the authored placeholders, but preserve the functional hierarchy and role metadata unless a step in this guide explicitly says otherwise.

2. Before you begin

- Make a separate development copy of your place.
- Publish the development place and enable HTTP requests in Game Settings > Security so the licensed runtime can verify normally during tests.
- Wait for AVCS verification to finish before judging vehicle initialization.
- Duplicate the correct conversion jig. Never edit the master jig in place.
- Rename the duplicate to the final unique template name immediately.
- Use one consistent name for the vehicle model, spawner configuration, and preview model.
- Build and test one subsystem at a time: authored structure, mobility, turret/gun, crew, combat, presentation, then spawners.
- Test with Start Server and at least two clients before release. Single-client Play mode cannot reveal every ownership, replication, or multicrew problem.

Do not copy `Pack`, `WLSig`, or `WLVer` attributes from an included production vehicle. Customer conversions are unsigned by design. A production signature describes an exact authored topology and must never be reused on a different rig.

3. The non-negotiable contract

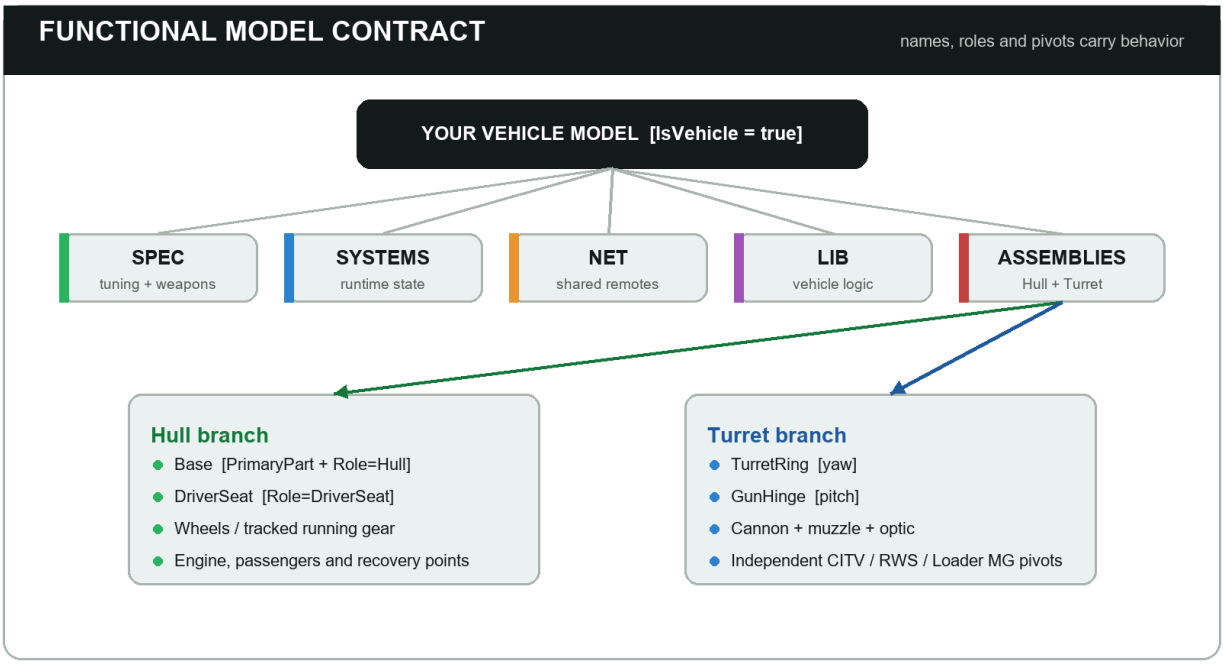

Figure 2. The minimum model hierarchy and the two load-bearing assembly branches.

Keep these items from the jig unless the documentation for a specific extension says to replace them:

- The root model attributes `IsVehicle`, `VehClass`, `ConversionJig`, and `UnsignedTemplate`.

- The `Hull`, `Turret`, `Systems`, `Net`, `Spec`, `DamageZones`, `Veh_Collision`, `Veh_Dmg`, `Veh_Xray`, `Lights`, and `JIG_GUIDE` containers that exist in the selected jig.
- `Hull.Base` as the model PrimaryPart and the part with `Role = Hull`.
- The driver `VehicleSeat` with `Role = DriverSeat`.
- `Turret.TraverseRing` with `Role = TurretRing`.
- `Turret.ElevationPivot` with `Role = GunHinge`.
- The `MainGun` assembly with `Role = Cannon`.
- Functional attachments, constraints, remotes, libraries, scripts, camera anchors, muzzle references, recovery equipment, and seat-loader objects.
- Existing `Role`, `Zone`, `Mount`, `MGStation`, `ModNode`, and `ModNodeOff` metadata unless you are deliberately configuring that feature.

The `JIG_GUIDE` folder contains ObjectValues pointing to the five most important references:

- `HullRoot`
- `DriverSeat`
- `TurretYawPivot`
- `GunPitchPivot`
- `WheelOrTrackRig`

If one of those ObjectValues becomes empty, stop and repair the reference before testing.

4. Duplicate and prepare the jig

1. Open `ServerStorage.AVCSConversionTemplates`.
2. Duplicate the tracked or wheeled jig.
3. Move the duplicate into a temporary conversion workspace folder or keep it beside the master while editing.
4. Rename it to the exact final template name, for example `My IFV`.
5. Set `DisplayName` to the customer-facing name.
6. Set `GunName` to the weapon name that should appear on the HUD.
7. Confirm `IsVehicle = true`.
8. Confirm `VehClass` still reads `Tracked` or `Wheeled` as appropriate.
9. Leave `UnsignedTemplate = true` and do not add protected pack/signature attributes.

The jig's visual language is intentional:

- Green envelope primitives are replaceable visual geometry.
- Blue volumes are collision or damage-related placement guides.
- Orange anchors are functional control or interaction references.

Replace visual envelopes. Reposition functional guides. Do not delete functional objects merely because they are invisible at runtime.

5. Scale, orientation, and PrimaryPart

- Keep the vehicle facing the same local forward direction as the jig.
- Do not rotate only the imported visual model and leave the functional rig facing another direction.
- Choose the vehicle's physical scale before placing suspension, armor, crew, cameras, damage zones, or muzzle references.
- Keep a consistent studs-per-meter value. The root `DisplayStudsPerMeter` is used by presentation and measurement systems.
- Preserve `Hull.Base` as PrimaryPart.
- Keep `Hull.Base` near the physical center of the hull assembly, with a stable orientation and a sensible center of mass.
- Do not use decorative mesh bounds as the main physics body. Use simple authored collision volumes.
- Avoid one enormous convex MeshPart collision hull. Several simple volumes are more predictable and cheaper.

For a pre-placed wheeled vehicle, author `Hull.Base.Anchored = true`. The wheeled initializer performs the controlled release. Do not pre-unanchor a wheeled vehicle and expect Workspace placement to remain deterministic.

6. Replace the visual geometry

Place hull-bound visuals in `Hull.Veh_Visual` and turret-bound visuals in `Turret.Veh_Visual` unless a dedicated moving assembly already exists.

The default replacement parts demonstrate the intended mounts:

- `JIG_HULL_LOWER_REPLACE_ME`, `JIG_HULL_UPPER_REPLACE_ME`, `JIG_ENGINE_DECK_REPLACE_ME`, and `JIG_DRIVER_HATCH_REPLACE_ME` are welded to `Hull.Base`.
- `JIG_TURRET_REPLACE_ME` and `JIG_COMMANDER_HATCH_REPLACE_ME` are welded to `Turret.TraverseRing`.
- `JIG_GUN_BARREL_REPLACE_ME` is welded to `Turret.ElevationPivot`.

For each imported visual part:

1. Place it at the correct location.
2. Parent it under the assembly that owns its motion.
3. Set `Anchored = false` unless it is a temporary edit-time guide.
4. Set `CanCollide = false`, `CanTouch = false`, and normally `CanQuery = false` for decorative geometry.

5. Set `Massless = true` for decorative parts.
6. Weld it to the correct functional root.
7. Delete the matching placeholder only after the replacement is mounted and verified.

Do not weld hull meshes to the turret, turret meshes to the gun, or optics to the wrong yaw/pitch pivot. A mesh can appear correct at rest and still orbit around the wrong axis at runtime.

7. Build the tracked running gear

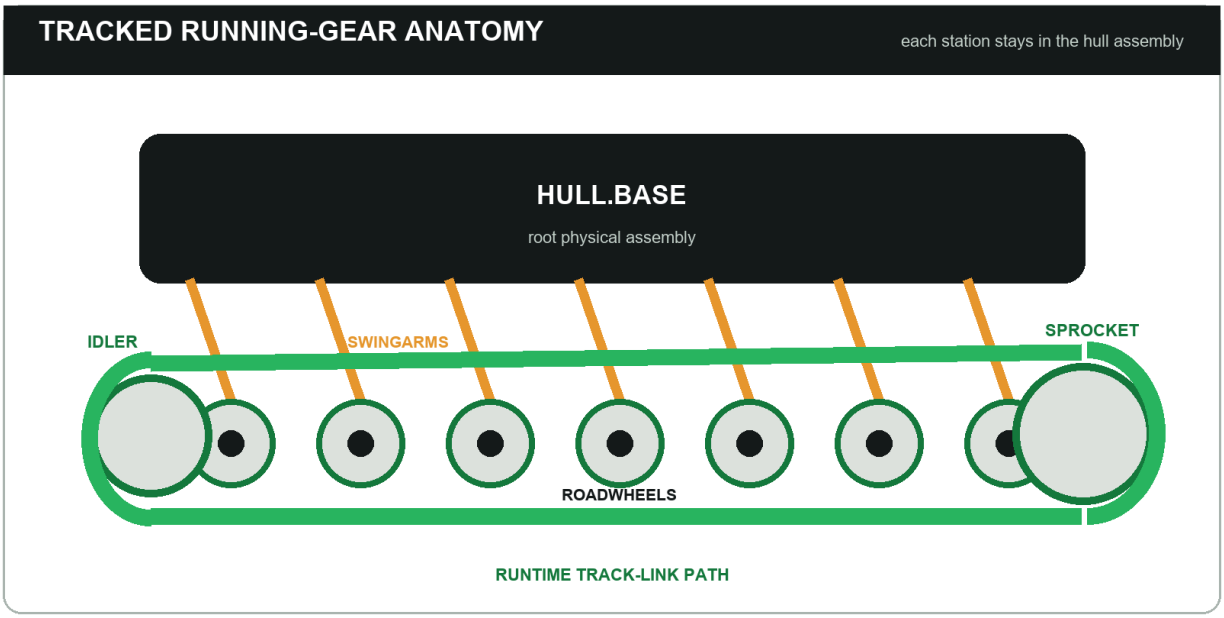

Figure 3. Tracked stations, swingarms and the runtime link path remain part of one hull assembly.

The tracked jig contains left and right suspension branches under `Hull.Suspension`.

Required organization

- Keep separate `Left` and `Right` sides.
- Keep the drive sprocket, idler, return rollers, and roadwheel bogies inside the correct side.
- Preserve each bogie's functional `AxleHub`, `Roller`, `Damper`, attachments, and constraints.
- Replace the `__SOURCE_GEOMETRY` wheel and swingarm visuals while retaining the functional roller/axle objects.
- Keep the side naming convention consistent. Do not swap left and right after animation tuning.

Roadwheels and swingarms

- Position each bogie's guide at the real wheel center.
- Move its damper attachments to represent the intended suspension axis and travel.

- Mount the roadwheel visual to the bogie's `Roller` through the existing wheel-spin weld pattern.
- Mount swingarm visuals to the moving suspension element, not to `Hull.Base`.
- Verify every roadwheel can travel without clipping the hull collider.

Sprocket, idler, and return rollers

- Position the sprocket at the true drive-wheel center and the idler at the true idler center.
- Mount their visuals to the supplied rotating roller objects.
- Return rollers should follow the authored top run of the track.
- Check rotation direction on both sides. Use the `Spec.Track` sign settings only after the geometry is correct.

Track links

- Use the included generic track-link asset or a customer-owned replacement with equivalent attachment logic.
- Configure `Spec.Track` values such as model, scale, count, connector offsets, lift, scroll sign, and spin sign.
- Keep generated links as presentation geometry; collision and traction should come from the controlled running-gear physics, not hundreds of colliding links.
- Test link spacing at full suspension compression and extension.

Tracked mobility tuning

Use `Spec.Mobility`, `Spec.Traction`, `Spec.Suspension`, `Spec.Weight`, and the related legacy values exposed by the jig. Tune one category at a time:

- Target mass and center of mass.
- Maximum forward and reverse speed.
- Acceleration and torque.
- Steering and neutral-steer rate.
- Brake force.
- Suspension stiffness, damping, and travel.
- Track grip and maximum climb angle.

Do not compensate for bad geometry with extreme torque. First verify the collision hull, wheel contact, suspension axis, mass distribution, and track-side assignment.

8. Build the wheeled running gear

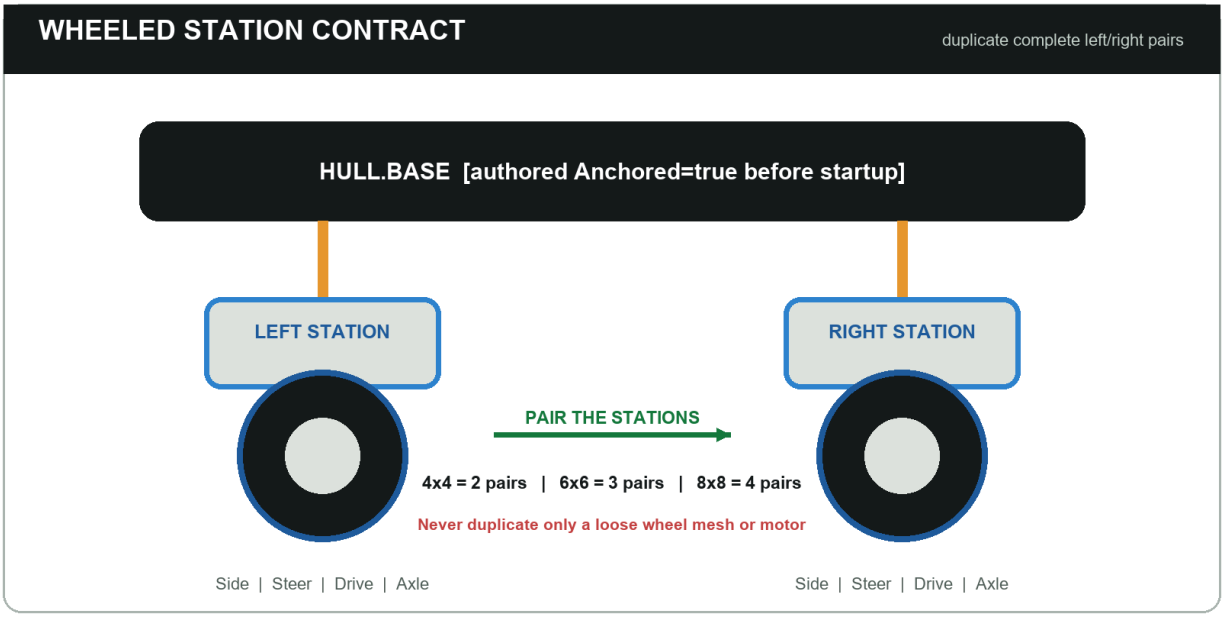

Figure 4. Build 4x4, 6x6 or 8x8 layouts by duplicating complete left/right station pairs.

The wheeled jig uses `Chassis.Wheels` plus `CombatWheelStub` and the wheel drivetrain scripts.

Wheel groups

- Keep each wheel in a dedicated group named with the `W_l_` or `W_r_` side convention.
- A 4x4 conversion can use two left and two right wheel groups.
- A 6x6 or 8x8 conversion can add the matching number of indexed groups.
- Do not assume the assist or drivetrain requires exactly eight wheels; it detects the configured wheel set.

Each group should retain the functional wheel part, hinge/attachment contract, and any steering pivot used by the jig. Replace only the source visual mesh once it is mounted to the functional wheel.

Drive and steering assignment

- Set the intended AWD, FWD, or RWD behavior in `Systems.WheelSettings`.
- Identify which axles steer and verify their steering sign.
- Verify left/right drive signs before changing acceleration or torque.
- Keep the wheel radius and suspension geometry consistent with the imported visual.
- Tune ride lift, steering decay, minimum steering, acceleration, deceleration, brake deceleration, brake force, and ignition timing through `WheelSettings`.

Small-obstacle assistance

`ReplicatedStorage.VehicleConfig.WheeledStepAssist` provides bounded low-speed assistance for small ledges. It supports straight and angled approaches and evaluates individual leading-wheel contact.

Tune the assist only after the vehicle climbs ordinary ramps correctly. Useful controls include enabled state, torque multiplier, maximum step-height ratio, maximum activation speed, minimum throttle, approach direction, probe rate, assist duration, and cooldown.

The assist is not intended to make a wheeled vehicle climb walls or compensate for a wheel collider that does not match the visible wheel.

9. Turret, gun, and sight pivots

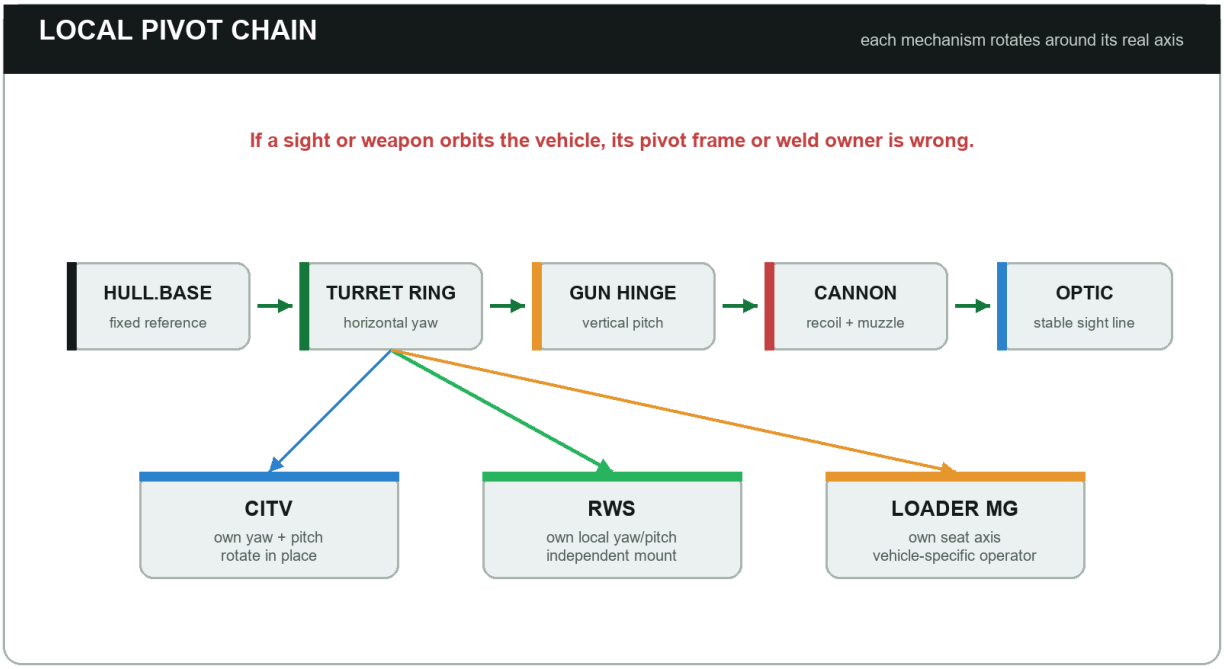

Figure 5. The turret, gun, optic and optional top-mounted systems require independent local pivots.

The gun system depends on two separate axes:

- `Turret.TraverseRing` controls horizontal rotation.
- `Turret.ElevationPivot` controls gun elevation and depression.

Position each pivot at the actual center of rotation before mounting the visuals.

Turret setup

- Mount the turret shell, turret hatches, turret equipment, and turret-owned crew seats to `TraverseRing`.
- Mount components that should stay with the hull outside the turret assembly.

- Do not solve a wrong pivot by offsetting the visible turret after it has been welded. Move the pivot and rebuild the local mount relationship.

Main gun setup

- Keep `Turret.MainGun` and `Role = Cannon`.
- Place `MuzzleRef` and `MuzzleEmit` at the physical muzzle.
- Keep `MuzzleEmit.GunMuzzlePoint` aligned with the bore direction.
- Place `GunSightCam` at the authored sight location.
- Place the coaxial `Muzzle` at the real coax position.
- Mount the visible barrel and any recoil-moving parts to `ElevationPivot` or the dedicated recoil assembly.
- Position the casing-ejection reference if the weapon ejects visible cases.

Turret specification

Use `Spec.Turret` for traverse rate, elevation rate, maximum elevation, maximum depression, and stabilizer strength. Use root/weapon recoil attributes and `Spec.Weapons.RecoilForce` for presentation. Test the gun at every limit and through a full 360-degree traverse where supported.

10. Commander optics, RWS, and Loader MG

Optional weapon/optic assemblies should have their own logical model instead of being mixed into the main turret shell.

CITV

- Create a dedicated `Turret.CITV` model.
- Use one horizontal pivot for the base and one vertical pivot for the sight head.
- Mount only the parts that belong to each axis.
- Place the CITV camera inside the moving sight head.
- Confirm the assembly rotates in place rather than orbiting around the main gun or turret center.

RWS and Loader MG

- Use separate models such as `Turret.RWS` and `Turret.LoaderMG`.
- Give each station a unique `MGStation` integer.
- Set the station's authored traverse and elevation rates.
- Keep its muzzle and sight references inside the station.
- If the station exposes a crew member, provide the role seat, hatch source, and exposed-seat transform.

- Decide whether the exposed player follows the turret only, rotates with the MG around the seat's own axis, or remains fixed relative to the turret. Do not weld the player seat to the gun barrel.
- Hatch state changes for entering or leaving the exposed MG position should be immediate unless the conversion deliberately supplies a tested animation.

11. Crew, seats, hatches, and exits

AVCS supports driver, gunner, commander, loader, and passenger roles. Only create roles that the vehicle actually carries.

Seat ownership

- Hull-fixed seats belong to the hull assembly.
- Turret crew seats must follow `TraverseRing`.
- A gun-mounted or top-MG exposed seat must follow the intended station without orbiting around an unrelated pivot.
- Passenger seats normally remain hull-fixed.

Entry prompts

- Keep the driver entry structure supplied by the jig.
- Place the entry anchor at the physical hatch.
- Assign the glow target to a visible authored hatch mesh, not an invisible helper part.
- Multiple roles may share one hatch through stacked prompts with distinct role labels/keybinds.
- Add passenger entry through the supplied passenger-hatch pattern where appropriate.

Exit positions

- Give each crew/passenger group a safe local exit reference.
- Place exit references beside the hull, not above the vehicle roof.
- Keep them clear of tracks, wheels, doors, walls, and the muzzle path.
- Do not reuse the driver's exit transform for every infantry seat.
- Test dismount while stationary, moving, beside a wall, under a roof, and while another player occupies the vehicle.

Multicrew rules

- Define only real crew stations.
- Mark manually loaded vehicles so an absent loader prevents reload.
- Vehicles without a loader station should use their authored automatic reload behavior.
- Confirm each role receives only its own HUD and controls.

- Confirm a passenger receives no driver/gunner UI.
- Confirm vacant stations can be selected through the existing Tab crew menu.

12. Power, fuel, and transmission data

Configure the root and `Spec` values appropriate to the selected jig.

Common categories include:

- Engine crank times and warm-start window.
- Fuel maximum, idle burn, throttle burn, low threshold, and refuel rate.
- Transmission gear limits and manual control keys.
- Acceleration, torque, maximum speed, reverse cap, steering rate, braking, and parking-brake behavior.
- Shift-jolt minimum speed, body kick, camera pitch/roll, duration, rebound, and rebound delay.

Keep visual jolt values moderate and test in both singleplayer and multicrew. A gear change must not create a different extreme camera impulse merely because another player owns the gunner or commander station.

13. Ammunition and weapons

Shells are defined under `Spec.ShellData` as `Configuration` objects. Give each shell a unique name and an `Order` value.

Typical shell attributes include:

- `kind`
- `round`
- `speed`
- `drop`
- `mass`
- `pen`
- `penDecay`
- `dmg`
- `humanDmg`
- `splash`
- `fragRadius`
- `explosiveKg`
- `airburst`
- `tracer`

Use `Spec.Weapons` for ammunition capacity, reload time, deload time, belt size, belt reload time, cannon rate of fire, coax rate of fire, top-MG rate of fire, smoke count, smoke reload, smoke duration, and recoil force as applicable.

Do not create a shell identity in a secondary system without adding its canonical profile to the shell/damage configuration. A fallback shell can produce completely unrelated penetration behavior.

After configuration, test:

- Every selectable shell.
- Empty chamber and reload.
- Manual-loader absence in multicrew.
- Autoloader or belt reload where fitted.
- Main gun, coax, RWS, Loader MG, autocannon, missile, and smoke authorization.
- Exposed infantry damage and armored-occupant protection.

14. Collision armor and damage zones

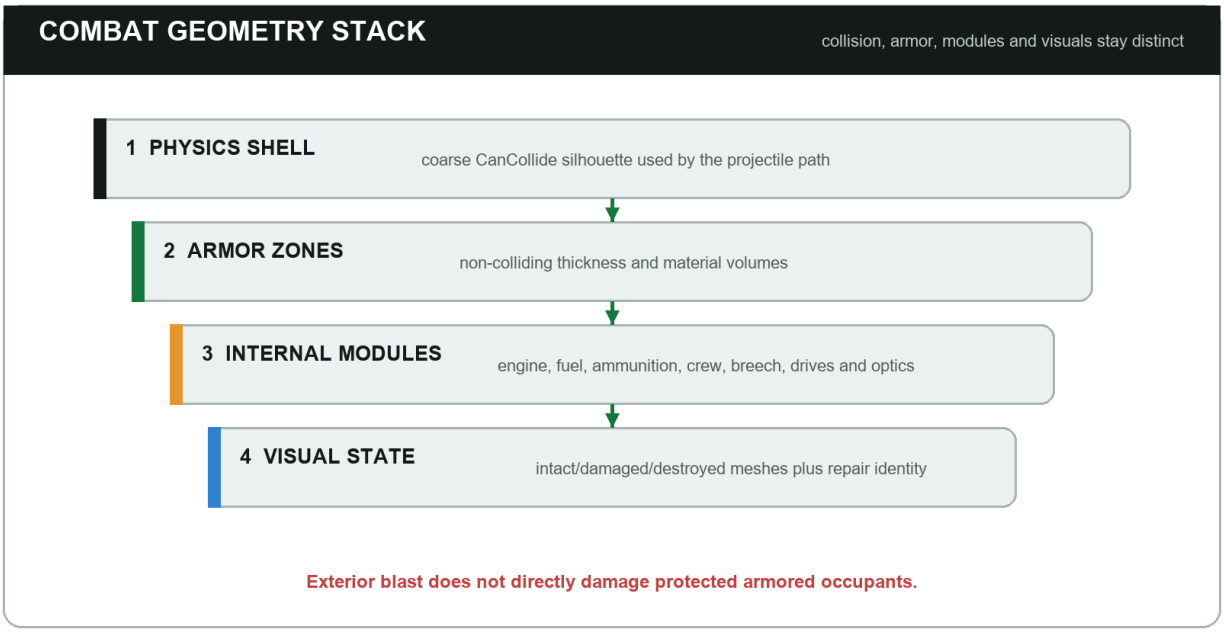

Figure 6. Keep projectile collision, armor, internal modules and visual damage as separate layers.

AVCS separates visible geometry from broad collision armor and internal damage zones.

Broad collision armor

Use `Veh_Collision` for simple armor volumes. The jig includes examples such as:

- `body_front_dm`
- `body_side_l_dm`

- `body_side_r_dm`
- `body_top_dm`
- `ex_armor_l_01_dm`
- `ex_armor_r_01_dm`

Match these volumes to the real protected area. Avoid fuzzy nearest-mesh assumptions. Use stable, deliberate identities.

Internal zones

The jig provides:

- `Zone_CREW`
- `Zone_ENGINE`
- `Zone_FUEL`
- `Zone_HULLAMMO`
- `Zone_BUSTLEAMMO`

Each carries a `Zone` attribute. Resize and position only the zones that exist in the vehicle. Delete or disable an inapplicable zone deliberately; do not leave a generic ammunition volume covering the entire interior.

Module placement

Place functional module hitboxes at their real locations and give them stable authored types/IDs. Common modules include barrel, breech, optics, horizontal drive, vertical drive, engine, transmission, fire-control equipment, tracks/wheels, ammunition, fuel, and crew.

Use simple collision geometry and verify the hit-camera X-ray model matches the damage model. If a hit camera shows a component in a different location from its damage zone, the conversion is not finished.

15. External damage panels and mesh swaps

`Veh_Dmg` contains damaged and destroyed visual variants. The example convention uses matching stable names:

- Normal: `ex_armor_l_01`
- Damaged: `ex_armor_l_01_dmg`
- Destroyed: `ex_armor_l_01_dstr`
- Collision: `ex_armor_l_01_dm`

Use a unique stable base identity for every repairable external panel. Do not depend on stripping `.001` suffixes or on nearest-mesh guessing, because several pools can otherwise resolve to one glow target.

For cosmetic panels that do not change vehicle function:

- Register them as external panels.
- Supply normal/damaged/destroyed variants if visible swapping is desired.
- Allow the grouped Exterior Panels repair operation to restore all damaged cosmetic panels.

For a functional panel such as reactive armor, a sight door, or a blowout panel, configure its specific behavior separately.

16. Protected ammunition and blowout panels

Only configure protected blowout behavior when the vehicle actually has separated protected ammunition storage.

- Place the bustle-ammunition zone inside the protected rack.
- Identify the panel meshes that should move.
- Place each hinge at the real panel hinge line.
- Set the outward/open direction.
- Place the fire-spout emitter at the opening through which pressure and flame should vent.
- Keep the normal closed mesh correctly aligned at runtime.
- Test the panel movement from multiple angles and after turret rotation.

An unprotected ammunition rack should use ordinary cookoff/detonation behavior. Do not enable a survivable blowout simply to make the vehicle harder to destroy.

17. X-ray and hit-camera geometry

Place internal presentation geometry in `Veh_Xray`. The jig includes engine, ammunition, and crew placeholders.

- Weld hull-fixed internals to `Hull.Base`.
- Weld turret internals to `TraverseRing`.
- Weld gun/breech internals that elevate with the weapon to `ElevationPivot` or the correct gun root.
- Give every X-ray item a stable component identity that matches the damage module.
- Tag modification-owned X-ray parts with the same modification logic as the visible part.
- Test with each modification both enabled and disabled.
- Keep X-ray geometry non-collidable, massless, and hidden outside authorized X-ray/hit-camera presentation.

The hit camera is only as accurate as the conversion data. Every component shown should correspond to a real damage pool and occupy the same space as its authoritative hitbox.

18. Repair, refuel, and ammunition service

Repair

- Keep the RepairKit workflow server-authoritative.
- Provide a usable exterior service anchor.
- Confirm each repairable module appears in the consolidated repair menu.
- Confirm all damaged external panels glow, and undamaged or unrelated meshes do not.
- Confirm every completed repair plays repair audio.

Refuel

- Keep a refuel service anchor near the engine/fuel access area.
- Configure fuel maximum and refuel rate.
- Enable the port from the Tab menu, equip `RefuelKit`, and hold R at the port.
- Test with engine off and running.
- Confirm the port closes when full and can be manually disabled.

Rearm

- Keep an ammunition service anchor near the appropriate loading point.
- Enable the port from the Tab menu, equip `AmmoKit`, and hold T.
- Confirm live shell/belt/missile counts are restored.
- Confirm the port closes after successful service.

19. Lights, audio, effects, and antennas

Lights

Replace and reposition the placeholders under `Lights`, including headlight and tail-light examples. Preserve the light-role mapping. Configure colors, blink rate, daytime running behavior, brake/reverse behavior, and whether tail lights follow headlights under `Spec.Lights`.

Audio

Use the vehicle's audio configuration modules for engine idle/effort layers, tracks/wheels, starter, weapons, turret movement, reload mechanisms, alarms, and damage states. Test as the driver and as an observer in another client.

Effects

Place muzzle, casing, exhaust, damage-fire, blowout, missile, and surface-effect emitters at their actual origins. Effects should be triggered by authoritative state and budgeted by the selected performance profile.

Antennas

- Parent hull antennas to the hull and turret antennas to the turret.
- Place the base at the real mounting point.
- Keep the antenna's environment probes lightweight.
- Test against the gun, a low bridge, nearby geometry, and while stationary.
- Verify the antenna settles at idle and that another client sees the approximate bend state.

20. Smoke launchers and missiles

Smoke launchers

- Parent launcher banks to the assembly that carries them.
- Set smoke count, salvo size, reload/cooldown, and duration.
- Point launch emitters away from the vehicle.
- Test every salvo until empty and after replenishment.

Guided-missile launcher

If the conversion carries an external deployable launcher:

- Create a dedicated launcher assembly and deployment pivot.
- Place loaded missile models inside the tubes in the stowed/deployed authored relationship.
- Attach missile visuals to the launcher until launch.
- Provide front and rear/cable reference parts for the missile flight model.
- Keep folded fins hidden in the tube and reveal them after launch as configured.
- Provide an exterior reload prompt and disable automatic reload if the design requires field loading.
- Test launcher deployment, stow, firing authorization, guidance, empty state, exterior reload, and multicrew operation.

21. Recovery points and towing

Use `Hull.RecoveryGear` and the supplied rear station as the pattern for additional front/rear stations.

- Place the prompt where a player can safely reach it.
- Place the cable/hook attachment on a structural point of the hull.
- Keep recovery parts mounted to `Hull.Base`.
- Set recovery stiffness, damping, reel rate, drag, braking, and boost values in `Spec.Recovery`.

- Test cable tow and rigid tow with tracked and wheeled partners.
- Test turning, reversing, slopes, detach, driver exit, and one vehicle being unoccupied.
- Confirm disabling Tow Winch in the Tab menu immediately detaches the active cable.

Do not add custom ownership changes inside the vehicle. Recovery authority is coordinated centrally.

22. Optional modifications

Spawner modifications are data-driven.

1. Add a `Mods` folder under the vehicle's spawner `Configuration`.
2. Add a BoolValue for each modification. Its Value is the default selected state.
3. On parts that exist only when the modification is enabled, set `ModNode = <ModName>`.
4. On base parts replaced by the modification, set `ModNodeOff = <ModName>`.
5. Apply the same tags to the full template and its preview model.

The spawned vehicle records the chosen state as `GarageMod_<ModName>`.

Apply modification tags to all dependent content: visible meshes, collision armor, X-ray geometry, damage zones, weapons, cameras, and effects. A disabled modification must not remain visible in X-ray or the hit camera.

23. Garage/depot registration

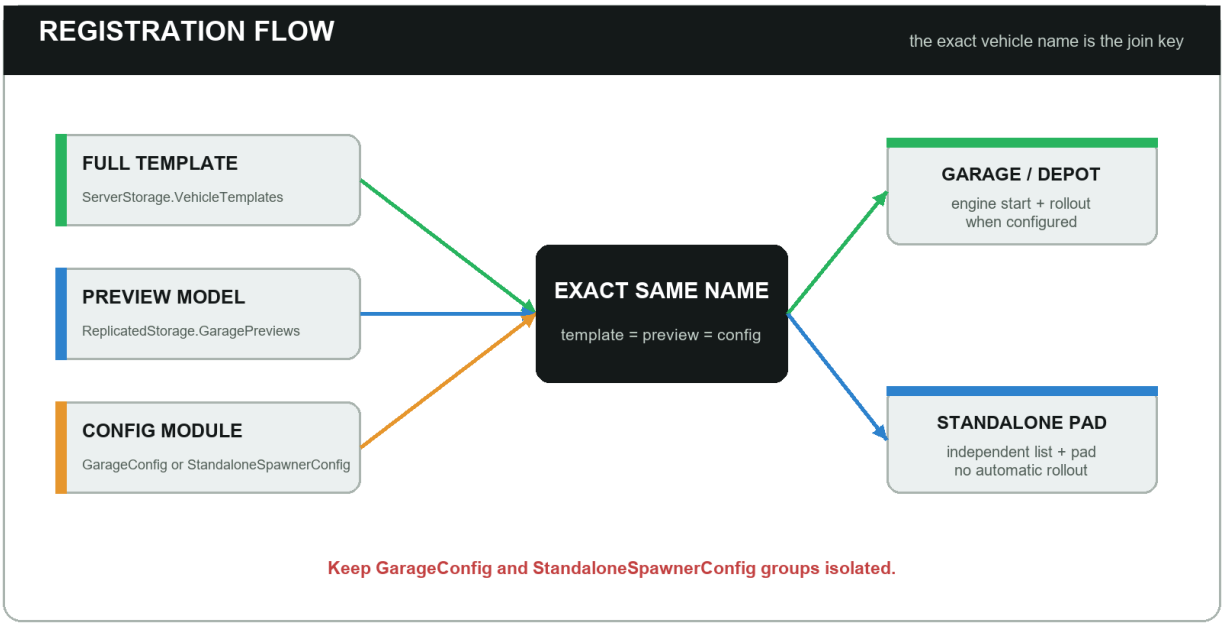

Figure 7. Template, preview and configuration use one exact name while garage and standalone groups remain independent.

The animated garage/depot requires the same exact name in three locations:

1. Full vehicle model: `ServerStorage.VehicleTemplates.<Name>`
2. Visual preview model: `ReplicatedStorage.GaragePreviews.<Name>`
3. Vehicle configuration: `ReplicatedStorage.GarageConfig.<Name>` or inside the intended team folder

Useful configuration attributes include:

- `Enabled`
- `DisplayName`
- `Class`
- `MinRank`
- `Team`
- `Order`

Root-level configurations are universal. A configuration inside a team-named folder belongs to that team. Garage-wide settings include team/group enforcement, GroupId, cooldown, per-player vehicle limit, automatic oldest replacement, and drive-out distance. Admin user IDs can bypass the locks.

The garage building can define its own team lock and each pad can override exit distance. Keep each bay's visible `DoorController` and follow its `HOW_TO_CONFIGURE` value for custom doors/slats.

After updating a vehicle, replace both the stored full template and its point-in-time preview clone.

24. Standalone spawner registration

Standalone pads use `ReplicatedStorage.StandaloneSpawnerConfig` and the same three-part template/preview/config naming rule.

- Register only the vehicles intended for that standalone group.
- Give each group its own system/group identifier.
- Do not point a standalone group at the garage roster unless shared access is deliberate.
- Set the physical occupancy region to cover only the pad and required spawn clearance.
- Test driving the vehicle completely clear, then spawning another without despawning the first.
- Test cancelling/despawning during the spawn transaction and confirm the reservation clears.

A standalone spawn does not start the engine or perform garage drive-out behavior.

25. Direct Workspace placement

AVCS supports placing a completed vehicle model directly in Workspace.

1. Start from the finished copy stored in `ServerStorage.VehicleTemplates`.

2. Duplicate it into Workspace.
3. Confirm `IsVehicle = true`.
4. Set `AVCSTemplateName` to the exact stored template/config name when the Workspace model name differs.
5. Set `CrewControlMode` to `Singleplayer` or `Multiplayer` if you do not want the default Singleplayer mode.
6. For a wheeled vehicle, confirm `Hull.Base` is authored anchored before runtime release.
7. Do not manually enable/disable individual vehicle scripts while startup preparation is running.

`VehiclePlacementStartup` verifies protected rigs where applicable, applies runtime-only rig fixes, creates the crew configuration, applies default modifications, welds managed camera anchors, clears residual velocities, marks the vehicle as pre-placed, and releases scripts/prompts through one startup barrier.

If startup fails, inspect `AVCSStartupState` and `AVCSStartupError` on the vehicle before changing the rig.

26. Performance profiles

`ReplicatedStorage.VehicleConfig.AVCSPerformanceProfiles` supplies four profiles:

- `Legacy` - highest update rates and minimal budgeting; useful for comparison or small controlled tests.
- `Cinematic` - high presentation quality with moderate filtering.
- `Balanced` - default profile for ordinary production use.
- `LargeBattle` - tighter remote budgets for larger servers while retaining full nearby presentation.

Profiles budget aim replication, guided-missile updates, vehicle audio, tracers, impact/muzzle effects, track presentation, projectile simulation, dormant physics, lights, and power updates. They do not transfer damage authority to the client.

Test the conversion in Balanced and LargeBattle. Nearby suspension, tracks, antennas, firing, and audio should remain responsive; distant work should scale down cleanly.

27. Validation

The jig's authored validation instruction is:

```
require(game.ReplicatedStorage.VehicleLib.Validate)(yourVehicle)
```

Run validation in a licensed test session after AVCS verification has completed. If the validator reports unauthorized before verification finishes, wait for the license state and run it again. Do not bypass validation by copying a production signature.

Validation is the beginning of testing, not the end. Also perform the following checks.

Edit-time structure check

- Model name, config name, and preview name match.
- `Hull.Base` is PrimaryPart and has `Role = Hull`.
- Driver seat has `Role = DriverSeat`.
- Traverse and elevation pivots have the correct roles.
- JIG_GUIDE references are populated.
- Visual parts are massless/non-collidable and welded to the correct owner.
- No required `Systems`, `Net`, `Spec`, or functional attachment was deleted.
- Customer conversion remains unsigned.

Mobility check

- Engine/battery start and stop correctly.
- Every forward and reverse gear works.
- Vehicle can steer at low and high speed.
- Braking and parking brake are distinct and controllable.
- Suspension settles without collapse.
- Tracked links, wheels, sprockets, idlers, and swingarms animate correctly.
- Wheeled steering/drive assignment and small-obstacle climbing work straight and at an angle.
- The vehicle can climb an ordinary ramp without extreme torque.

Gunnery check

- Turret rotates around its own center.
- Gun elevates around the trunnion.
- Muzzle, camera, reticle, laser, and projectile path agree.
- Every shell can be selected and fired.
- Reload/chamber behavior matches the loader arrangement.
- Coax, top MG, smoke, and missile systems work only where fitted.
- Hit-camera feedback names the actual damaged components.

Crew/network check

- Singleplayer and Multiplayer spawn modes both initialize.
- Each crew role receives only its own controls/HUD.
- An empty required station becomes unavailable without AI backfill.
- Crew switching only allows vacant fitted stations.
- Entry glow uses the correct hatch.

- Every seat exits at its own safe location.
- Driver and passenger exits do not stop or corrupt the drivetrain.
- Another client sees smooth hull, suspension, turret, gun, antenna, audio, and damage state.

Damage/logistics check

- Armor zones correspond to the visible vehicle.
- Engine, fuel, ammunition, crew, tracks/wheels, breech, barrel, optics, and drives can be damaged where fitted.
- Armored occupants are protected from exterior splash.
- Exposed occupants can be damaged.
- Repair menu lists real damaged components and all damaged panel glows.
- Refuel and ammo ports require the equipped matching kit.
- Fire suppression, cookoff, blowout, destruction, and repair effects clear correctly.

Spawner check

- Preview matches the full vehicle and selected modifications.
- Garage doors/startup/drive-out occur in order.
- Standalone spawn remains independent and does not start the engine.
- Occupancy clears when the vehicle leaves the physical region.
- Interrupted/despawned spawn reservations clear.
- Direct Workspace placement reaches `AVCSStartupState = Ready`.

28. Troubleshooting

Vehicle is denied as unlicensed or tampered

- If it is an included production vehicle, restore the pristine signed template and do not change authored weld/Motor6D/WeldConstraint/attachment topology.
- If it is a customer conversion, remove copied `Pack`, `WLSig`, and `WLVer` attributes and return to the unsigned jig workflow.
- Confirm HTTP requests are enabled and verification has completed.

Hull moves but running gear stays behind

- Check every suspension/drive part is connected to the intended assembly.
- Look for anchored descendants.
- Confirm no custom script changes network ownership.
- Confirm the model initialized through the startup barrier.

Turret or CITV orbits instead of rotating in place

- The moving visuals are mounted to the wrong pivot or the pivot is not at the real axis.
- Rebuild the local mount relationship. Do not offset the camera to hide the rig error.

Tracks/wheels spin while stationary

- Confirm visual spin follows actual vehicle displacement.
- Check for scripts that drive presentation from stale throttle rather than motion.
- Confirm the vehicle is not receiving a retained drive intent.

Repair/refuel/rearm prompt completes but state does not change

- Equip the exact required tool.
- Confirm the port is enabled and the resource is not already full.
- Check server output and authoritative result state.
- Confirm the component identity in the repair menu matches the actual damage pool.

Standalone pad remains occupied

- Check for an active spawn reservation.
- Inspect the physical occupancy region for invisible vehicle parts.
- Confirm the vehicle has fully left the region.
- Confirm the standalone group and pad identifiers are unique.

Pre-placed wheeled vehicle collapses at startup

- Author `Hull.Base.Anchored = true` before runtime.
- Check for other anchored descendants and broken wheel constraints.
- Let the managed wheeled initializer release the assembly.

29. Final release checklist

- Keep the master conversion jigs unchanged in `ServerStorage.AVCSConversionTemplates`.
- Store the completed vehicle in `ServerStorage.VehicleTemplates` and its matching preview in `ReplicatedStorage.GaragePreviews`.
- Register the exact name in the intended GarageConfig and/or StandaloneSpawnerConfig group.
- Confirm customer conversions carry no protected pack/signature attributes, then run validation after license initialization.
- In a multi-client server, test singleplayer, multicrew, direct placement, both spawners, combat, repair, logistics, towing, and performance profiles before publishing.
- Keep a clean local backup of the finished template and configuration.