Runtime kda25 / Converter V2.0 kda12 | Updated 2026-10-28

01 Read first: role and working contract

For the AI assisting a KDA user. This technical companion supports the beginner Prompts guide; its examples are not requests to implement every feature. Current version: runtime V1.0 / 2026-10-28-kda25; car-only converter V2.0 / 2026-10-26-kda12 with original hybrid audio. Check actual source, hashes and the current user request before relying on a historical description.

Establish access and intent

State what repository/files you can actually read. A private GitHub URL or a local path in chat does not itself grant access. With only PDFs you can explain design, but need relevant source to edit it. Ask only for essential missing inputs, and do useful inspection meanwhile. Read README.md and applicable AGENTS.md. Establish car, intended behavior and target KSP installation.

Translate simple requests into a narrow subsystem change. Do not ask beginners to select algorithms, source classes or telemetry columns. Explain the plan and next action in plain language. If access is limited, say so rather than pretending to inspect or modify files.

Respect explicit scope

Update converter code, EXE, source snapshots or portable packages only when explicitly requested. A game runtime/material request alone does not authorize a converter rebuild. Diagnose first, then clarify any necessary converter change. Do not retune Race, Drift or engine braking unless the request actually targets that behavior.

Build and check a game update first. Do not ask the user to close KSP. Report ready. When installation is requested, verify KSP is closed and install with backup/hash verification without a second closure confirmation. If KSP is running or cannot be verified, leave the update ready. Establish target installations rather than assuming everyone has both main and test copies. Preserve exports, craft and saves.

Report evidence honestly

Distinguish observed facts from hypotheses, built from installed, and static validation from native driving. A screenshot or one run rarely proves a unique cause. Proceed with clear, authorized work without unnecessary approval loops. Return what changed, what was checked, remaining uncertainty and a short next test.

02 Repository, dependencies and build inputs

Location	Purpose
GameData/KDAVehicles	KDA runtime, crew, effects and Lizard GT and Rabbit Rally x as an included car.
Converter/KDAVehicleConverter.exe	Offline Windows converter with black UI and visible build indicator.
source/plugin	C# modules, pure cores and focused regression/replay tests.
source/converter/kdaconvert	Parsing, calibration, geometry/material/audio conversion, validation and installation.
source/.build/runtime/GameData/KDAVehicles	Ignored staged build output; release assets exist only in the top-level GameData.
source/tools	Build, model and documentation tools; vendored MU writer is a converter library.
documentation	Game Systems, SC, Converter, beginner Prompts, this AI Context and scoped update supplements.

External KSP mods are not bundled. ModuleManager 4.2.3 and HarmonyKSP 2.2.1.0 are required for KDA part/crew mass patches. Optional PBR visuals use Shabby 0.4.2 and Resurfaced 1.2. Download links/tested versions are in README.md and dependencies.json. FAR, AtmosphereAutopilot and KSP Community Fixes may affect a user installation but are not KDA dependencies.

Use the current repository as source of truth. GameData is the sole installable runtime tree, source/plugin is runtime code, and source/converter is the independent car-only converter. Build output belongs under ignored .build directories.

See source/README.md for builds. C# requires the user's installed KSP/Unity references; do not distribute them. The converter contains Python libraries and the GPL MU writer, not other KSP mods. Preserve retained asset/library notices, KDA-only GameData and only the requested demo car. Packaging uses release-manifest.json and source/tools/package_kda.py.

03 Native wheels, suspension and limits

A generated chassis has one rigid body and four independently selectable corner wheel parts. Native KSP wheel contacts provide physical suspension, tire/contact forces, steering, motor and brakes. The body owns engine/controller/aero/HUD modules. Decorative spinning meshes are not substitutes for wheel contact, and cosmetic art must not introduce duplicate physical contacts.

KDAModules.cs defines KDAEngine, KDAWheelMotor and KDAGrip. KDASuspension.cs and SuspensionCore.cs own springs/dampers/anti-roll; KDAAxle.cs links axle settings; KDAWheelSelection.cs provides editor selection. Verify class definitions before editing, rather than deriving file paths from class names.

Standard hardware retains 0.16 m travel. Rally/Desert kits implement extended travel with compression/droop, rest sag, progressive springs, damping limits, track geometry and visual anchors. Ride height alone is not additional travel. Preserve Standard behavior when modifying these systems and test rest, full compression, droop and landings.

Editor chassis presets and flight SC modes are separate. Road/Sport/Race chassis grip multipliers are 0.75 / 1.15 / 2.00 atop imported calibration; the slider caps at 3x. Source compound/width mapping is approximate, not BeamNG contact-patch simulation. Do not infer identical base grip across cars or translate a slider directly into measured tire friction.

Engine braking currently requests wheel drag through the native brake-input path. It is not a complete negative driveline-torque model. Preserve this behavior unless it is explicitly in scope. engine_drag, brake_input and aid demand can help separate causes, but native timing and source paths must be checked.

Spinning wheel geometry removes imported camber/toe to avoid wobble, preserves axle roll and mirrors rims outward. Keep tire radius, transforms, native steering ranges and selection ownership intact. Visual-only changes must not silently change mass, contact geometry or physics calibration.

04 Drive-by-wire modes and driver intent

Read KDAController.cs, KDACore.cs, RoadSportCore.cs, the SC manual and relevant replay tests. Mode IDs are 0 Road, 1 Sport, 2 Race, 3 Drift. SC is predictive keyboard drive-by-wire with slip/yaw recovery; it is not a heading autopilot, artificial sideways force or simply an integral PID.

Ordinary turn-in has a speed-dependent lateral budget. Recovery countersteer has separate authority so stock geometry reduction cannot starve high-speed correction. The runtime bypasses stock automatic steering reduction where needed. Compare command, normalized limits, actual wheel angle, native servo state and response; later actuation saturation can resemble weak SC.

Road/Sport use Race-derived predictive recovery with separate state and more conservative corner/throttle profiles. Race permits some slip; TCS does not guarantee zero oversteer. Drift preserves natural oversteer and integrates tap frequency/held-key trim progressively instead of instant full override. Neutral settling, sign transitions and release behavior matter to opposite-side oscillation.

Cornering allowance remains active for ordinary corner demand. It is not a measurement of unused grip and reducing it can hide a recovery defect. SC assistance, allowed oversteer, cornering allowance and chassis grip are separate controls. Track changes and output, not slider labels alone.

Drift SC lock is 25-75 degrees, default 50, with up to 15 extra degrees for keyboard input within physical steering range. TCS is forced off in Drift; automatic SC yaw brakes are disabled there. B uses front service brakes in Drift, both axles otherwise. Rear handbrake is a separate custom action-group hold binding; releasing the physical key clears it.

SC and TCS are separate. Outside Drift, SC can request yaw brakes and TCS can cut torque/request traction braking. There is no separate yaw-brake-only switch and no implemented GT3 1-10 Race intervention HUD control. Do not invent these as existing features. An explicit request for a new control may authorize adding it; document the changed behavior.

05 Engine, AWD, aero and audio contracts

Read EngineCore.cs, GearboxCore.cs, InductionCore.cs, DrivelineCore.cs and KDAModules.cs alongside export calibration. Imported transmission and induction types are fixed to the export. Gear ratios, final drive, torque curve and tire radius determine behavior; metadata top speed is not a hard speed cap.

Boost calibration separates base torque from source pressure ratio and reapplies boost once. Do not multiply an already boosted curve again. Turbo lag estimates use exported pressure/rotor/inertia data, not BeamNG Lua execution. Imported superchargers use calibrated boost without turbo lag. TwinCharged uses combined full-load output plus a bounded mechanical share and handoff; read Twincharged Imports.md/tests.

AWD source axle shares normalize across driven wheels before differential redistribution. A 40% front / 60% rear car nominally allocates 20% per front wheel and 30% per rear. A wheel's 100% motor limiter is its local allowance, not total engine torque applied independently four times. Open/Mild/Sport/Drift LSD change sharing; compare wheel speeds, slip, load and torque.

KDAAero scales signed axle reference forces by density and forward-speed squared, applies them at imported points and converts N to KSP rigidbody kN. Reverse/inverted attitude reduces downforce. Negative downforce means lift. Aero does not guarantee contact over a jump or rough ground. Check load, contact and surface before proposing artificial adhesion.

HybridAudioCore.cs and KDAHybridAudio.cs implement original shared RPM/load banks, live intake/boost/shift layers and bounded playback. KDAAudio.cs and KDASampleAudio.cs preserve saved module names and route to that backend. No exported engine recordings are used. Voice Auto resolves engine layout. Revision 5 uses deeper, family-specific generated banks and baked cycle variation. Saved tone/mix values remain internal; only volume/exhaust are exposed in the garage, and volume in flight. Maximum twelve combustion layers/car and four audible hybrid cars; paused, packed, muted, map and warp states stop playback. Managed tests do not prove native FPS or audio quality. Request a short recording for audible faults; telemetry records engine states, not sound waveforms.

06 Converter, materials and visual articulation

Input is an Automation BeamNG ZIP export, not a standalone .car, arbitrary BeamNG car or official Automation plugin. Disable Merge All Fixtures for separately identifiable details; merged exports can still convert with reduced separation. Embedded .car metadata may supply engine information. Inspect the selected config, JBeam slots/variables, DAE, material JSON and DDS maps; do not infer engine or paint from names.

In source/converter/kdaconvert: service.py orchestrates atomic output; automation.py/characteristics.py/engine_metadata.py inspect and calibrate; mesh.py writes geometry through the MU library; materials.py/resurfaced.py translate materials; suspension_visual.py/exhaust_visual.py find visual bindings/outlets; ksp.py writes config/craft/audio; wheel_config.py generates independent wheel configs; validation.py/installer.py check and install packages.

Use safe extraction and the restricted numeric JBeam expression adapter; do not execute untrusted source expressions or Lua. Preserve torque/pressure/ratio/drivetrain/aero data where supported and report missing data honestly. An unsupported export error should lead to inspection of that ZIP, not a fabricated default engine or silent omission.

Material pipeline 7 handles DAE inputs both labeled set="0" despite distinct -UV0/-UV1 source IDs. Repeating detail UVs and opacity stamps remain separate. Keep body/fixture alpha cutouts. Recognized camso_t_grille<digits>_h_stepped.dds materials get perforation alpha and a dedicated atlas; ordinary dark paint must stay opaque. See Grille Materials.md. Masked PBR preserves alpha and averages smoothness over visible pixels.

For shinier/more matte requests, identify the actual material, source maps and shader path; adjust roughness/metalness/specular without losing color, normal detail or alpha. Wrong UVs, missing cutout alpha and excess smoothness need different fixes. Compare source, converted maps and native KSP rendering. Editing a screenshot is not a game material fix.

Visual suspension uses selected JBeam groups/anchors and native wheel travel with a lightweight cosmetic skin. Unsupported layouts remain rigid; Lizard GT rear art is a known example. Retain chassis, subframe, exhaust and unsupported suspension geometry. Do not invent missing articulation anchors, add physical joints/colliders for cosmetic animation, or remove unsupported components.

Installer guards include existing output, traversal/redirects, running KSP, duplicate/mismatched runtimes and incomplete packages, with scoped backup/rollback. A conversion is not an installation. External PBR mods are not installed by the converter. Supporting a new export in the converter is an explicit converter request; a current-car visual request alone is not.

07 Crew and choosing the right evidence

KDACrew.dll is separate from the vehicle physics DLL. KDACrew.cs owns seats/doorways; KDAInteriorShell.cs supplies inward-facing cosmetic coverage. Imported bodies have no hidden crew/command after the ModuleManager patch. A real occupied command seat provides control, portraits and native IVA. Keep an outward, clear doorway marker on the matching seat channel for EVA/boarding, with native obstruction checks.

The interior shell reverses selected source faces, keeps cutouts, skips windows and adds matte dark floor/firewall coverage. It has no colliders and is hidden outside IVA. It cannot invent missing cockpit art. Internal and external Kerbal visuals share the lowered seating position. Third-party first-person mod compatibility is not certified.

Symptom	Useful evidence
Handling, oscillation, wheelspin or braking	Whole telemetry run; car, mode, surface, intended maneuver and key behavior near a MARK.
Quiet/clipping/broken sound	Short video/audio, rpm/load situation and KSP volume settings; telemetry if timing matters.
Wrong texture, holes, missing visual part	Screenshot, original Automation BeamNG ZIP, converter build and installed shader stack.
Conversion error	Original ZIP, complete error/session log, selected settings and converter build.
Cannot drive, board or EVA	Part/crew screenshot, module/dependency state and KSP.log around the failure.

Guide beginners through one useful test. Do not request a technical spreadsheet when setup.cfg captures much of it. Ask what the car should have done. If one car is targeted, avoid a global retune. Explain whether existing craft use the change immediately or whether regenerated models/configs are needed.

08 Telemetry collection, files and timing

Read KDATelemetry.cs, TelemetryCsv.cs and the run README.txt. Current schema is 4. KDAHud.cs exposes RECORD / STOP RECORDING and MARK in flight. Begin needs an active, unpacked car. Folder: <active KSP root>/KDATelemetry/<local timestamp>-<vessel name>-<short id>/. Main/test roots are independent.

Coach the user: choose car/mode/settings; RECORD; check REC/sample count; drive normally briefly; reproduce one issue; MARK near it; STOP RECORDING; allow the writer to finish; ZIP the entire run folder and describe intended behavior. MARK stores time and sample count, not video or a description. Do not require straight-only or Road-only runs without a diagnostic reason.

File	Contents
setup.cfg	Schema/runtime SHA, vessel/mass/timestep, part configs, current module fields, wheel labels/radius and native friction fields.
vehicle.csv	Vehicle body state, keys/axes, SC outputs, aids, engine/gear/boost/aero and diagnostics for each physics sample.
wheels.csv	Available wheel rows per sample: contact, steering, slip/load/forces, drive/brake/aid/grip states.
settings.csv	Initial module fields, then changes polled at 1 Hz; core SC flight settings also appear in vehicle.csv.
events.csv	START, MARK and STOP with elapsed time and details.
summary.txt and README.txt	Final stop reason, dropped queued packets, writer errors; schema, timing and unit caveats.

Sampling occurs after SC/recovery actuation in FixedUpdate. Native contact/force state can be a step older than the command. sample is run-local; time_s is elapsed game seconds; ut is universal time; dt_ut is actual game-time delta; fixed_dt is nominal physics step. Do not assume a fixed real-world video clock.

Recording stops on vessel/scene changes, packing, part-count changes, sampling/writer errors or shutdown. A bounded background queue reports dropped packets, not necessarily complete missing samples. summary.txt appears after worker completion; a missing summary may mean copying too early or abrupt termination. Inspect the files before declaring corruption.

09 Telemetry fields, units and joins

Read actual CSV headers; older logs differ. Join vehicle.csv to wheels.csv on sample within one run, retaining wheel part_id. setup.cfg maps IDs to corners. Apply settings.csv/events.csv by time_s, allowing 1 Hz polling uncertainty. Missing native wheel readings are omitted rows, not measured zero load. Use a real CSV parser: strings may be quoted and contain commas/newlines.

Field semantics matter: sc_active is the observer's confirmed-slide state, not simply SC enabled. tcs_active means TCS is enabled and allowed in that mode, not proof that torque is being cut. Use per-wheel tcs_torque_scale and aid_brake_pressure to inspect intervention. With SC present, raw_steer already includes steering trim; do not add raw_steer_trim twice.

Group	Fields and interpretation
Inputs / timing	sample, time_s, dt_ut, key_a/d/w/s, raw_steer/trim, raw_throttle/trim, brakes, handbrake_active. Key columns are literal A/D/W/S; remapped input must also be interpreted through raw axes.
Body state	speed_kmh, forward/lateral/vertical_mps, slip_deg, yaw_rad_s, yaw_accel_rad_s2, grounded_wheels, roll_ground_deg. Local x right, y forward, z down; reported vertical/up acceleration follows explicit source signs.
Controller	sc_mode/enabled/active, allowed_oversteer, cornering_percent/budget_g, turn_limit, recovery_limit, driver_steer, sc_correction, output_steer, filtered_slip/yaw. Normalized limits and steering outputs are not degrees.
Prediction / settling	filtered_slip_rate_deg_s, filtered_yaw_rate_rad_s2, desired_yaw, race_recovery_bias, race_turn_in_scale, race_settling_weight, drift_bias/lock, sc_throttle_scale, sc_yaw_brake. Check source units.
Wheel / contact	actual_steer_deg, max_steer_deg, command_steer, servo_steer, stock_steer_deg/input, auto_steering, response, grounded, omega_rad_s, slip_x/y, load, contact_depth, suspension_compression.
Torque / aids / grip	drive_torque, brake_torque, drive_input, brake_input, aid_brake_pressure, abs_scale, engine_drag, tcs_torque_scale, grip_slider, friction_multiplier/adherent/peak/sliding. Native force/torque units are verbatim; verify API conventions.
Engine / aero	rpm, gear, selector, requested_throttle, engine_throttle, torque_nm, boost_bar, turbo_rotor, engine_load, shifting, limiter, aero_downforce_n. torque_nm is instantaneous, not rated peak.

Mass is tonnes; aero force is N; speed is named km/h or m/s; slip is degrees; yaw is rad/s; CSV booleans use 0/1. Do not deduce corrective direction from a key alone. Compare orientation, velocity, measured yaw/slip, wheel angle and the rig convention.

10 Diagnostic reasoning and verification

Handling investigation

Check runtime SHA/schema, summary/events, sample ranges and wheel joins first. Reconstruct settings; separate grounded driving, takeoff/landing, impacts, burnout/handbrake and mode transitions. Treat MARK as a time window. Compare key onset/release, slip/yaw and derivatives, filtered state, limits, correction, actual steering, wheel load and torque/braking, allowing native contact-state lag.

Large requested correction but small actual wheel angle can indicate actuation/range constraints. Opposite-side overshoot after release can involve filtering, sign prediction or settling. Sustained drift with wheelspin may involve throttle, differential or load. These are hypotheses to test, not automatic diagnoses. Lower cornering allowance can conceal a recovery issue. Use matched runs and change one factor when useful.

Minimum useful checks

For controls, use focused pure-core/replay checks and native comparison runs where available: hold/release, tapping, countersteer and both transition directions. For travel, test sag, compression/droop and landing without changing unrelated modes. For materials, inspect source/baked maps and render the actual asset. For converter edits, validate a fresh package and disposable install/rollback paths, retaining source metadata and notices.

Keep KDA-only packaging and the requested demo car. Record versions/hashes and distinguish the build on disk from the DLL actually loaded during a run. Static package validation is not a driving test. Do not claim to have installed, driven, opened or heard something you did not verify.

Return to simple language

Explain what happened, the best-supported cause, what changed or remains uncertain, and one short next test. Teach RECORD / MARK / STOP and folder selection without requiring CSV expertise. Put technical detail in this context and source evidence, not in every prompt a beginner has to paste.

11 Current runtime and converter architecture

Runtime 2026-10-28-kda25 and converter V2.0 / 2026-10-26-kda12 are separate products in the current release sources. Read Hybrid Audio.md, Converter.md and the garage/suspension guides for the current feature set. The tested runtime is kda25; check actual installed hashes instead of assuming an older stable copy. release-manifest.json inventories actual download bytes; release-manifest.json records package file hashes.

LongTravelCore/KDALongTravel implement Standard/Rally/Desert hardware, sag targets, progressive resistance, bounded damping, track settings and suspension telemetry. KDAGarageTravel and KDALongTravelVisual provide editor tuning and visual articulation. Saved rates change only when the user applies settings, a calculator or a preset. Tune / Tires controls front/rear KDAGrip independently of chassis presets.

KDANitrous and NitrousCore implement opt-in rechargeable bottles, shot size, held key/action input and grounded drift recharge. KDAHud and KDAHudLauncher provide driving controls and the stock flight-toolbar toggle. AfterfireStyleCore and KDAAfterfire supply revised exhaust signals and a low-volume Flame plume. KDAExhaust gives blue nitrous precedence over orange fire. Keep these cosmetic systems separate from engine torque except for explicit nitrous power.

KDAWorldGarage uses the KDAGarage catalog and KDAEntrance transform; assets are under GameData/KDAGarageBuilding/Garage. It owns overview entry, complete craft handoff, player session/recovery files and door-aligned driveway launch. KDAWorkshopRoom renders the separate textured editor interior. KK and Custom Pre Launch Checks are optional external dependencies, never bundled.

The converter makes car-only packages and never contains or replaces runtime/crew/garage assets. It verifies the installed base ABI, minimum build, exact runtime hash and crew prerequisites; preserves scoped backups and rollback; then installs only the car and craft. Five-part output remains uncrewed until a physical seat/doorway and Kerbal are assigned. Native BeamNG vehicle-mod conversion is unsupported.

Current source has no duplicate GameData snapshot. Runtime/crew builds stage under source/.build/runtime; model rebuilds stage under .build/assets. Build tools take explicit KSP paths for external game assemblies. Preserve current module IDs, exact saved craft identity and player data unless the requested change includes a migration. The generic placement migration tool writes a separate output and retains UUID/pose/scale.

12 Current accepted behavior and scope

Runtime kda25 adds ExtremeSpeedGovernor for assisted imported Race/Sport cars, fading in from 65 to 80 m/s (234-288 km/h). At full weight, driver turn-in remains within the cornering budget and automatic yaw correction is capped at half that limit. Large slip-alignment recovery and automatic yaw-brake feedback fade out; electronic axle vectoring offsets are limited to three percentage points and yaw feedback fades out. Total drive allocation is conserved. Road, Drift, ordinary-speed calibration, suspension, grip and the separate rover ECU are unchanged. This trades large-slide automatic recovery for bounded intervention at extreme speed.

The separate rover ECU supports Unstable and Normal only. It is an add-on aid: do not add engine conversion, Fun mode, power changes, suspension changes or grip changes. Exactly two front wheels steer while enabled; disabling restores original steering settings. Its cornering allowance is a percentage of the calculated budget. Consult Rover ECU.md and source/ecu for the authoritative implementation.

Do not rebuild the converter for runtime-only or documentation-only work. Converter remains V2.0 / 2026-10-26-kda12; runtime is kda25. Third-party assets retain their own terms.