**LalaLaunch – working state & what’s next (handover)**

**Repo / branches**

* Main repo: **LalaLaunchPlugin**
* Recent working branches we touched:
  + centralize-track-resolution (track key/name resolver shared by Fuel + Profiles)
  + fix/profiles-auto-refresh (Profiles tab refresh + selection logic)
  + fix/fuel-load-lookup / fix/fuel-nonlive-save-key (Fuel tab load/save against track **Key**)
  + fix-pit-stop-results → fix-pit-stop-clean-up (pit loss math + dash publishing)
* Current local work is on **fix-pit-stop-clean-up** (then merged as needed).

**Big changes (landed)**

1. **Track identity unified**
   * Added CarProfile.ResolveTrackByNameOrKey(...) and used it from:
     + Fuel: FuelCalcs.ResolveSelectedTrackStats()
     + Profiles: ProfilesManagerViewModel.EnsureCarTrack(...), PB updates, and UI refresh.
   * Result: no more “unknown/duplicates”, Fuel & Profiles agree on the same track instance.
2. **Profiles tab live refresh**
   * After we create/ensure a track (live or replay), the Profiles tab:
     + Refreshes its track list
     + Re-selects the newly touched track
   * Eliminated the “click away and back” workaround.
3. **PB helper hardened**
   * PB update path now prefers **track key**, then falls back to resolver.
   * Lap-time validity checks + min/max guards in place.
   * “PB source” kept simple: profile vs live.
4. **Pit‐stop time loss logic**
   * We now compute two things:
     + **Direct Pit Lane Travel** (stopwatch, entry→exit): for a quick baseline.
     + **DTL (race pace delta)** = (In + Out) − 2 × AvgLap (no subtraction of box time here; it’s the “rival stayed on track” loss).
   * **Save rule:** on a valid cycle we save **DTL** to profile; if DTL is missing/≤0, save **Direct** instead (rounded seconds). Debounced so it saves once.
   * Works with pit box **before/after** S/F; verified at Daytona (both variants) and matches manual timing to within ~1s.
5. **SimHub properties (dash)**
   * Exposed a focused set under the plugin (browse “LalaLaunch” → “Pit.\*” in SimHub):
     + Pit.DirectTravelSec (stopwatch)
     + Pit.DTLTotalSec (race pace delta)
     + Pit.AvgPaceUsedSec + Pit.AvgPaceSource (“profile-avg” → “live-median” → “session-pb”)
     + Pit.InLapSec, Pit.OutLapSec, Pit.DeltaInSec, Pit.DeltaOutSec
     + Pit.Debug.TimeOnPitRoad (counts while on pit road, then freezes)
     + Pit.Debug.LastPitStopDuration **now counts in stall and freezes on exit** (we pointed this at PitEngine.CurrentPitStopElapsedSec)
     + Pit.CandidateSavedSec, Pit.CandidateSource (what got persisted to profile)
   * A trimmed **TestingData–Pit Stop Info** dash uses only these and renders a clean sequence for In/Out/DTL.

**What to expect in logs**

* PitEngine: Direct Pit Lane Travel Time calculated: {sec}s
* PitEngine: Pace Delta Time Loss calculated: Total={dtl}s (avg={avg}, in={in}, out={out}, stop={stop})
* LalaLaunch: Mined new PitLaneTimeLoss ({rounded}s) and updated profile/UI.  
  (appears once per valid cycle)

**Known quirks (non-blocking)**

* We still publish the derived fields one lap later in a few sequences if the “immediate publish” block doesn’t run (e.g., weird replay ticks). Most runs now publish right after out-lap S/F; if not, they appear after the next S/F.
* Dash property names are standardized under Pit.\* going forward (older “Lala.\*” groups will be retired later).

**Test recipe (replay or live)**

1. Start at least one clean lap to seed averages.
2. Enter pits → stop in box → exit pits → complete the out-lap.
3. Watch properties:
   * While on pit road: Debug.TimeOnPitRoad counts; in box: Debug.LastPitStopDuration counts; both freeze appropriately.
   * After out-lap S/F: InLapSec, OutLapSec, DTLTotalSec, CandidateSavedSec should be populated & frozen until next cycle.
4. Confirm profile shows the saved **DTL** for the car/track; Fuel tab picks it up on reload.

**Backlog (captured for next passes)**

* **Profiles hygiene:**
  + If profile DTL is 0/default: auto-fill on first valid stop.
  + Else: show “new candidate {sec}s vs saved {sec}s” with a **[Accept]** button near the field.
  + Add **[Reset to default]** for DTL and later for tyre-change/refuel rate mining.
* **Immediate publish UX** (finish polish):
  + Ensure InLapSec/OutLapSec/DTLTotalSec are always snapped at **out-lap** S/F and remain frozen until next pit event.
* **Delegates naming pass:**
  + Migrate everything to Pit.\* / Fuel.\* groups (no redundant “Lala.”), keep old keys temporarily for backward compatibility, then prune.
* **Fuel tab PB button**: honour the same average-source precedence and lap guards as pit math.
* **General clean-up:** remove commented/unused vars noted; keep logging at “Info” for pit events, “Debug” elsewhere.

**Code pointers (where to look next time)**

* **Pit math & timers:** PitEngine.cs
  + LastDirectTravelTime, LastTotalPitCycleTimeLoss
  + CurrentPitStopElapsedSec (new): returns running or latched stall time
  + State machine for pit phases + lap finalization
* **Live tick + publishing:** LalaLaunch.cs
  + UpdateLiveFuelCalcs(...) – lap crossing logic, average selection, debug fields, immediate-publish snap, and attach delegates
  + Pit\_OnValidPitStopTimeLossCalculated(...) – debounce + candidate selection + profile/UI update
* **Profiles & tracks:**
  + CarProfiles.cs – ResolveTrackByNameOrKey(...)
  + ProfilesManagerViewModel.cs – EnsureCarTrack(...), PB updates
  + FuelCalcs.cs – ResolveSelectedTrackStats()

**How to start next work session**

1. In VS: Git → **Checkout** main, then **Create new branch** from main (suggested: pit/immediate-publish-polish).
2. Build, run with your Daytona replay and the **TestingData–Pit Stop Info** dash.
3. If you want a fixed baseline, set **Avg Lap** in the profile first (we prefer **profile-avg**, then live-median, then PB).

When you open the next chat, paste this block and say which item you want to tackle first (I recommend the *Immediate publish UX* polish so the dash updates right at out-lap S/F every time).