Lean Hogs Perpetual Futures — Pricing, Oracles, and Risk Controls (CFD or CME)

This document consolidates the design for offering **Lean Hogs (LH) perpetual futures** either from exchange futures (CME HE) or, when exchange data is unavailable, from a **CFD feed (API Ninjas)**. It explains the Aug 15 "price crash," proposes robust oracle constructions, funding, and operational guardrails, and provides concrete examples and snippets compatible with Autonom-style feeds.

---

1) Why the Aug 15 drop wasn't a crash

- The series you looked at (**LH1**) is a **front-month continuous** chart.
- On **Aug 15, 2025** it **rolled** from the expiring Aug-25 contract to Oct-25. The next contract was ~18% lower, producing an apparent "crash" that is actually a **splice gap**.
- Moral: **continuous front-month charts embed roll gaps**. Perps should **not** target these naively; use a **constant-maturity index** or a CFD index with risk controls.

---

2) Oracle options

You have two viable anchors:

Option A — Constant-Maturity Futures (CMF) Index (when CME futures are available)

Target a fixed maturity (e.g., **30 days**) by blending the **front** (F_1) and **next** (F_2) contracts by time to expiry.

- Let $\mathbf{T_1}$ = days to expiry of F_1, $\mathbf{T_2}$ = days to expiry of F_2, and **T*** = target days (30).
- Weight on the front:

$$w = \frac{T_2 - T^*}{T_2 - T_1}$$

- Index:

$$P_{\mathrm{CMF}}(t) = w\,F_1(t) + (1 - w)\,F_2(t)$$

- **Properties:**
- Micro-rolls continuously (no day-one gap at roll).
- Hedgeable with **w** units of front and **(1–w)** units of next.

Illustrative example (numbers are examples): - Date: **Aug 10**; expiring Aug contract has $\mathbf{T_1 = 4}$ days; Oct has $\mathbf{T_2 = 66}$ days; **T*** = 30. - $\mathbf{w = (66-30)/(66-4) = 36/62 \approx 0.581}$. - If $\mathbf{F_1 = 109.50¢/lb}$, $\mathbf{F_2 = 90.40¢/lb}$ then: **P_CMF ≈ 0.581×109.50 + 0.419×90.40 ≈ 100.9¢/lb.** - As days pass, **w** ↓ and the index glides toward the next contract—no jump on roll day.

Optional blend: Close to expiry you can blend a small weight to the **CME Lean Hog Index (LHI)** for additional cash anchoring. Keep the weight modest except around final settlement windows.

Option B — CFD-Backed Index (when you only have API Ninjas)

Treat the **CFD Last** as your reference price but **harden** it.

1) **Pull & validate** (per-tick): - Reject stale: `now − updated > 90s` → hold last good. - Reject non-finite values; require monotonic timestamps. - Bound step changes: $|\Delta| \leq 5\%$ per tick (outside → hold last good + flag). - Accept updates only inside your configured **trading-hours window** (mirrors exchange hours), excluding the maintenance break.

2) **Stabilize**: - Publish **30–60s TWAP** of accepted prints as `LH-CFD-INDEX`. - Keep a **5–10 min rolling median** for spike suppression (used only when TWAP deviates beyond a threshold).

3) **Roll awareness**: - CFDs embed roll gaps. **Do not smooth** the price across a roll; that breaks hedgeability. - Mitigate with **temporary margin hikes** and **funding caps** during the expected roll window.

4) **Publishing**: - Sign each tick; include `{ price:int, expo:int, timestamp:int, stale:bool, source:string }`.

---

3) Marking & funding

Use the index (CMF or CFD) as the **funding target** and as the base for the **mark price**.

- **Mark price:** short TWAP (e.g., 30s) of the chosen index.
- **Funding every Δ (e.g., 8h):**

$$\text{Funding}_{\Delta} = \text{clamp}\left(\kappa \cdot \frac{P_{\text{mark}} - P_{\text{index}}}{P_{\text{index}}},\ -f_{\max},\ f_{\max}\right)$$

- **Tuning:**
- κ : sensitivity (start small for non-storable commodities).
- **Caps** $f_{\max}$: high enough to pull back spreads, low enough to avoid liquidation cascades on CFD rolls.

Concrete funding example: - Suppose `P_index = 0.90125` (90.125¢), `P_mark = 0.91000` (0.97% rich). - Choose $\kappa = 0.5$, $f_{\max} = 0.004$ (0.40% per 8h). - Raw funding = 0.5 × 0.0097 ≈ **0.00485** → **clamped to 0.004 (0.40%)** paid by longs to shorts.

---

4) Risk & market-ops controls

A. Roll / limit-up-down events - If the index prints stall (limit moves, CFD freeze): - Freeze index updates or widen TWAP window. - Temporarily **raise initial/maintenance margin** and **lower max leverage**. - **Throttle funding** (reduce κ, lower **caps**) until normal prints resume.

B. Trading-hours guardrails - Only accept/produce index updates inside configured **market hours**; publish `stale:true` outside.

C. Circuit breakers - If $|\Delta|$ > **8%** in < 60s **and** not in roll window → pause matching or switch to last-good mark.

D. Fallbacks - If feed is stale > **N minutes**: allow position **reductions only**; block new opens.

---

5) Data model & API shapes (Autonom-style)

5.1 Feed payload

```
{
  "feed_id": 143,
  "symbol": "LH-CFD-INDEX",
  "price": 9012500000,
  "expo": -8,
  "timestamp": 1755338221181,
  "stale": false,
  "meta": {
    "source": "api-ninjas/commodityprice?name=lean_hogs",
    "window_sec": 30,
    "twap": true
  },
  "signature": "<hex>",
  "recovery_id": 0
}
```

5.2 Batch response (example)

```
{
  "prices": [
    { "feed_id": 143, "price": 9012500000, "expo": -8, "timestamp":
1755338221181 }
  ],
  "errors": [],
  "signature": "<hex>",
```

```
  "recovery_id": 1
}
```

5.3 Funding parameters (per-asset)

```
{
  "asset": "LH",
  "funding_interval_hours": 8,
  "kappa": 0.5,
  "funding_cap": 0.004,
  "mark_twap_sec": 30,
  "use_index": "LH-CFD-INDEX"
}
```

---

6) Implementation examples

6.1 CFD oracle loop (Rust-style pseudocode)

```
use std::collections::VecDeque;
use serde::Deserialize;
use std::time::{Duration, SystemTime};

#[derive(Deserialize)]
struct NinjasResp { price: f64, updated:
i64 } // price in USD per lb (decimal), updated in ms

struct Twap { win_ms: i64, q: VecDeque<(i64, f64)>, sum: f64 }
impl Twap {
    fn new(win_ms: i64) -> Self { Self { win_ms, q: VecDeque::new(), sum:
0.0 } }
    fn push(&mut self, ts: i64, px: f64) {
        self.q.push_back((ts, px));
        self.sum += px;
        while let Some((old_ts, old_px)) = self.q.front().copied() {
            if ts - old_ts > self.win_ms { self.q.pop_front(); self.sum -=
old_px; } else { break; }
        }
    }
    fn value(&self) -> Option<f64> { if self.q.is_empty() { None } else {
Some(self.sum / self.q.len() as f64) } }
}

fn accept_tick(last_px: f64, last_ts: i64, px: f64, ts: i64, now_ms: i64) ->
```

```
bool {
    if !px.is_finite() || ts <= last_ts { return false; }
    if now_ms - ts > 90_000 { return false; }
    let step = if last_px > 0.0 { (px - last_px).abs() / last_px } else { 0.0 };
    step <= 0.05
}

fn encode_autonom(price: f64, ts: i64) -> (i64, i32) {
    // expose as integer + expo=-8, e.g., 90.125 → 9012500000
    let expo = -8; let int_px = (price * 10f64.powi(-expo)).round() as i64;
(int_px, expo)
}
```

6.2 Constant-maturity weights (Rust-style)

```
fn cmf_weight(t1_days: f64, t2_days: f64, t_star: f64) -> f64 {
    ((t2_days - t_star) / (t2_days - t1_days)).clamp(0.0, 1.0)
}
fn cmf_index(f1: f64, f2: f64, w: f64) -> f64 { w * f1 + (1.0 - w) * f2 }
```

6.3 Funding calculation (exchange side; Python)

```
def funding_rate(mark, index, kappa=0.5, cap=0.004):
    raw = kappa * (mark - index) / index
    return max(min(raw, cap), -cap)

# example
print(funding_rate(mark=0.91000, index=0.90125, kappa=0.5, cap=0.004))  # →
0.004
```

---

7) Ops playbook for rolls

1) **T–5 to T–1**: raise IM by +25–50%; tighten leverage; pre-announce funding caps. 2) **T–1 to T+1**: widen mark TWAP to 60–120s; throttle funding (lower κ , cap $\leq$ 0.25%/8h); enable circuit breaker at 6–8%/min. 3) **T+2**: revert parameters to baseline.

---

8) Backtesting notes

- Use a **back-adjusted** history (for analytics only) to estimate risk/funding; do **not** use it as a live index.

- Stress with synthetic roll gaps of 10–20% and limit-up/limit-down days; verify liquidation and funding paths remain stable.

---

9) Glossary

- **Front-month continuous (1st nearby):** Series that switches to the next contract at roll; introduces gaps when curve is sloped.
- **Back-adjusted:** Historical series normalized to remove roll gaps; good for charts, **not** for live pricing.
- **Constant-maturity index:** Blends contracts so the time-to-maturity is fixed; eliminates day-one roll gaps.
- **CFD index:** Uses a CFD price stream as the reference; requires sanity checks and roll-aware risk controls.

---

10) Quick “recipes”

CFD-only perp - Index = 30s TWAP of API Ninjas Lean Hogs CFD (validated + hours-gated). - Mark = 30s TWAP of index. Funding every 8h with κ =0.5, cap=0.4%/8h. - Roll window guardrails + circuit breakers + stale-mode.

CME-powered perp - Index = 30-day CMF (F_1/F_2 blend). Optional low-weight cash anchor (LHI) near expiry. - Same mark/funding engine and ops playbook.

Appendix A — Example end-to-end flow (CFD)

1. Pull `price, updated` from CFD → validate.
2. Push into 30s TWAP → produce `LH-CFD-INDEX` tick.
3. Exchange marks positions with 30s TWAP of index.
4. Every 8h: compute funding to target index; apply caps.
5. During roll windows: raise IM, lower caps, widen TWAP; revert after T+2.

Appendix B — Example end-to-end flow (CMF)

1. Pull **F_1/F_2** quotes + expiry dates → compute `w` each tick.
2. Compute `P_CMF = w·F₁ + (1−w)·F₂` → 30s TWAP.
3. Same steps 3–5 as CFD.