

# The Full Map: Tokenized Whisky Cask Infrastructure

## 1. THE REALITY OF CASK DATA

What data actually exists for a whisky cask?

CASK LIFECYCLE	DATA GENERATED	SOURCE	CONFIDENCE
Distillation	New make spirit gauge (volume, proof, date)	DSP records (TTB)	Hard fact
Filling	Entry gauge: volume, proof, cask serial, cask type, fill date, warehouse ID	Warehouse records (TTB-required)	Hard fact
Years 0-6 (maturation)	NOTHING NEW. Cask sits. No measurement. Angel's share is happening but nobody's measuring it.	-	-
Year 6+ (every 2-3yr)	Regauge (if owner requests): RLA, ABV, bulk litres, date, method	Warehouse (on request)	Imprecise Wet dip = in a barr ±5-10% var Full disgo accurate b
Transfer	Re-gauge at origin + destination New warehouse ID, date	Both warehouses (TTB-required)	Hard fact (gauges i
Bottling decision	Cask selected, final gauge	DSP/warehouse	Hard fact
Bottling	Cask emptied, final yield, bottles produced, proof	DSP records (TTB-required)	Hard fact
Sale/Transfer of ownership	Ownership transfer record (warehouse receipt assignment)	Legal docs	Hard fact

Key insight: There are massive data gaps during maturation.

Between fill and first regauge (potentially 6+ years), the ONLY data is: - Entry gauge (one measurement from years ago) - Mathematical estimate of angel's share (model, not measurement) - The cask physically exists (warehouse can confirm)

Between regauges (2-3 year intervals after year 6): - Last regauge data (stale within months) - Angel's share estimate since last regauge

### **Regauge accuracy is honestly poor for wet dip:**

- Cask sizes vary  $\pm 10\%$  from nominal (a "250L hogshead" could be 240-270L)
  - Temperature affects density readings
  - Operator technique matters
  - ABV measurement has decimal-place variability
  - Only full disgorge (emptying the cask) is truly accurate, and that's rare
- 

## **2. THREE DATA TIERS**

---

### **Tier 1: Verifiable Facts**

Things the warehouse/TTB records prove. Ground truth.

Data Point	Source	Update Frequency	Accuracy
Cask exists	Warehouse inventory	Continuous	Binary — yes/no
Cask type	Entry records	Once (immutable)	Exact
Fill date	Entry records	Once (immutable)	Exact
Entry gauge (volume, proof)	TTB-required gauge	Once	High
Location (warehouse ID)	Warehouse system	On transfer	Exact
Ownership	Warehouse receipt	On transfer	Exact (legal doc)
Last regauge data	Regauge report	Every 2-3yr (after yr 6)	Moderate (wet dip) to High (disgorge)
Lifecycle state	Warehouse records	On event	Exact
Bottling yield	DSP records	Once (at bottling)	Exact

## Tier 2: Computed Estimates

Math applied to Tier 1 data. Transparent methodology, clearly labeled.

Data Point	Inputs	Method	Accuracy
Current volume estimate	Last gauge + time elapsed + climate rate	Exponential decay model	±10-15% between regauges
Current age	Fill date + now	Subtraction	Exact
Estimated bottle yield	Volume estimate + cask type + quality factor	Linear model	±15-20%
Angel's share to date	Entry gauge + age + climate rate	Compound annual loss	±5% cumulative

## Tier 3: Market Opinions

Models, comps, vibes. Useful as reference, not truth.

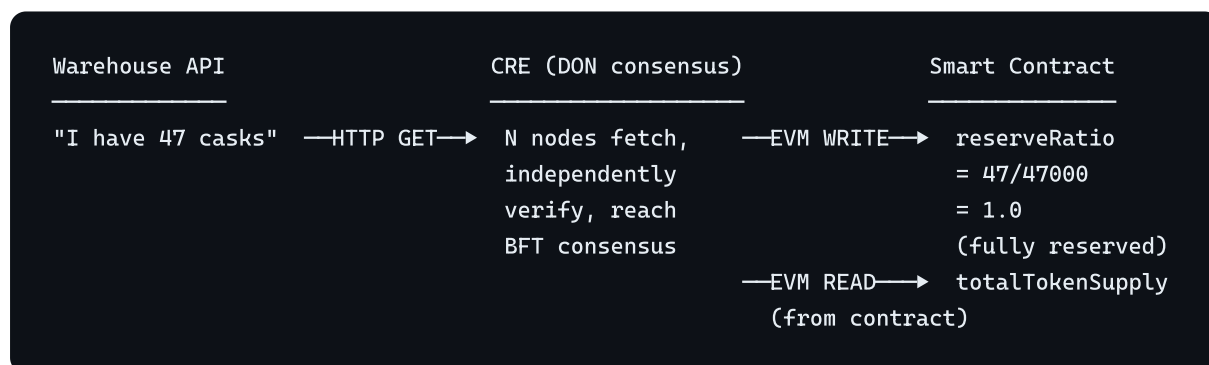
Data Point	Source	Reliability
Comparable auction prices	Auction houses (sparse, inconsistent)	Low-moderate (thin market)
Whisky index values	RW101, Knight Frank (quarterly)	Moderate (aggregate, not per-cask)
Age-based appreciation curve	Our model (calibrated to indices)	It's a model. Labeled as such.
Cask type premium	Market convention + auction data	Directionally correct, imprecise
"What is this cask worth"	All of the above blended	An informed opinion, not a price

### 3. WHAT WE ACTUALLY BUILD (GROUNDED)

#### Component 1: Proof of Reserve (Tier 1 only — strongest component)

**What it proves:** The number of physical casks in the warehouse  $\geq$  outstanding token supply.

**Data flow:**



**Why CRE matters here (specifically):** - Without CRE: Tim's server says "47 casks." Tim also issued the tokens. Self-attestation. - With CRE: N independent nodes fetch the warehouse data, independently verify, reach consensus. The attestation is "the DON confirms the warehouse reports 47 casks." Still trusts the warehouse (which is TTB-regulated), but the PIPELINE is decentralized. - With Confidential HTTP: Same attestation, but the actual number "47" never appears onchain or in node logs. Only "casks  $\geq$  tokens" boolean. Competitors can't extract inventory intelligence.

**Frequency:** Hourly (or configurable). This is checking inventory count, not measuring liquid.

**Honest limitation:** This proves cask COUNT matches tokens. It does NOT prove what's IN the casks. A warehouse could have 47 empty barrels. The trust model is: TTB-regulated

warehouse + CRE-verified data pipeline. Not trustless end-to-end.

## Component 2: Physical Attribute Oracle (Tier 1 + Tier 2)

**Rename from "Dynamic NAV Oracle." This is what it actually is.**

### What it puts onchain:

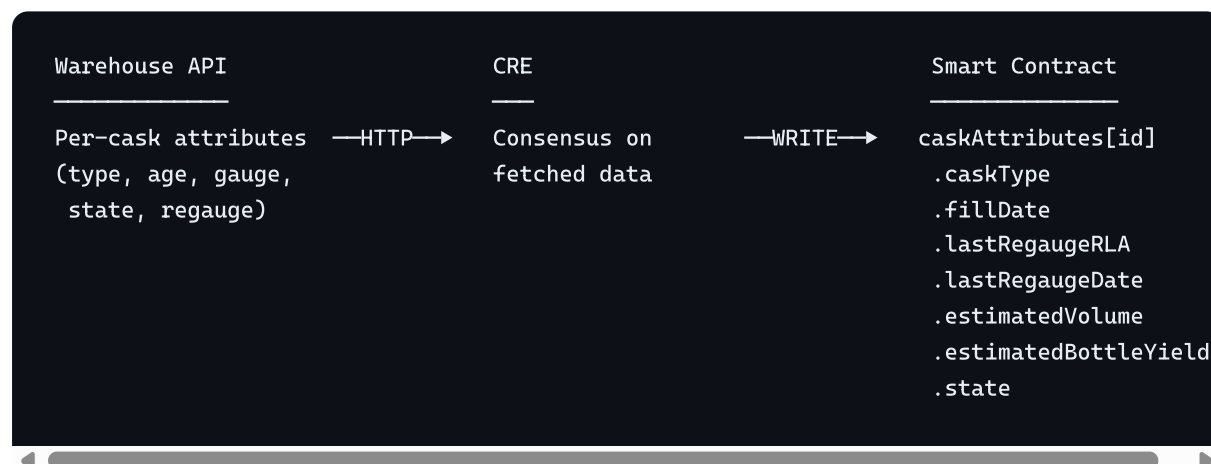
Per cask (from Tier 1 — warehouse records): - Cask type, fill date, entry gauge, last regauge data + date, warehouse ID, lifecycle state

Computed (Tier 2 — transparent math): - Current age (trivial) - Estimated current volume (angel's share model since last gauge, with model parameters visible) - Estimated bottle yield

Aggregate: - Total casks, total estimated volume, portfolio age distribution

**What it does NOT put onchain:** - A dollar value. Not our job to say what a cask is "worth." - Or if it does, it's CLEARLY labeled as "reference valuation model v1, methodology X, not a market price."

### Data flow:



**Why this matters:** - Any protocol or investor can read verified physical attributes - A lending protocol uses these to set their OWN collateral haircut - A secondary market prices tokens based on visible attributes - Nobody is forced to trust OUR valuation model — they have the data to build their own

**Frequency:** Daily or on-regauge-event. Attributes don't change fast (age increments daily, volume estimate drifts slowly).

## Component 3: Lifecycle Provenance (Tier 1 only)

**What it records:** Every state transition as an immutable onchain event.

```
fill → maturation → regauge → maturation → regauge → bottling_ready → bottled
```

Each transition logged with: cask ID, from state, to state, timestamp, metadata (gauge data if regauge, yield if bottled, etc.)

#### Data flow:



**Why this matters:** - Immutable chain of custody from fill to bottle - Solves the "receipts problem" — prove your whisky is what you say it is - Every regauge result is permanently recorded (can see volume trajectory over time) - Regulatory compliance: auditable trail of every event

**Frequency:** Event-driven (webhook from warehouse) + daily cron fallback to catch missed events.

### Component 4: Privacy Layer (Feb 12+)

**What changes:** Proof of Reserve inventory fetch becomes Confidential HTTP.

**What this enables:** - Prove "cask count  $\geq$  token supply" WITHOUT revealing: - Exact number of casks - Which warehouses - Barrel acquisition costs - Supplier relationships - All commercially sensitive. Competitors, other buyers, market participants shouldn't see raw inventory data.

**What doesn't need privacy:** - Lifecycle events (provenance = transparency by design) - Physical attributes of YOUR casks (token holders should see what they own) - Age, type, state (not commercially sensitive)

## 4. THE DATA SOURCE PROBLEM (HONEST)

The entire system depends on ONE data source: the warehouse API.

**Current state (hackathon):** - We build a FastAPI service that serves seeded mock data - CRE workflows fetch from it - This demonstrates the architecture

**What makes it credible anyway:** - Warehouse operators are TTB-regulated (US) / HMRC-WOWGR regulated (UK) - Falsifying records is a federal offense - The warehouse's business model depends on reputation (custody is their product) - This isn't random API data — it's legally mandated records

**Future evolution (beyond hackathon — mention in demo, don't build):** - Multiple independent data sources (warehouse A + warehouse B + independent auditor) - CRE can fetch from all three and require 2-of-3 agreement - IoT sensors (weight, temperature, humidity) as additional verification - Third-party regauge services as independent confirmation - This is an architecture that SCALES to more trust sources. Hackathon = 1 source. Production = N sources.

## 5. STAKEHOLDER VALUE MAP

### Cask Owner / Token Issuer (Tim)

Problem Today	What We Solve
Investors can't verify casks exist	Continuous proof of reserve, independently verified
Quarterly PDF appraisals are slow and expensive	Real-time physical attributes onchain
No provenance trail beyond paper records	Immutable lifecycle events from fill to bottle
Sharing inventory data exposes commercial info	Confidential HTTP: prove reserves without revealing details
High cost of capital due to opacity	Transparent verified data → investors accept lower risk premium

### Fund Manager / Institutional Investor

Problem Today	What We Solve
Stale NAV between annual appraisals	Continuous attribute data — build your own model on verified inputs
Expensive custody audits	Proof of reserve replaces point-in-time audits
Chain-of-custody documentation for compliance	Lifecycle events satisfy audit trail requirements
Can't independently verify broker claims	Read the contract. It's all there.

## DeFi Protocol (lending, DEX, etc.)

Problem Today	What We Solve
Can't accept RWA collateral without trusted price feed	Physical attribute oracle provides inputs for collateral models
No proof that collateral exists	Proof of reserve is composable — any protocol can read it
No standard for RWA data onchain	Standardized struct for cask attributes that any protocol can consume

## Warehouse Operator

Problem Today	What We Solve
Manual reporting to multiple clients/investors	Report once to API, oracle distributes
Liability for valuation claims	We don't publish valuations. Just facts. Warehouse not opining on price.
Competitive risk from sharing data	Confidential HTTP: data verified without public exposure

## 6. SMART CONTRACT REVISION

Based on this grounded mapping, the contract should store:

**Per-Cask Attributes (replaces "NAVData")**



```

struct CaskAttributes {
    CaskType  caskType;           // bourbon_barrel, sherry_butt, etc.
    uint256   fillDate;          // unix timestamp
    uint256   entryGaugeVolume;  // original volume in mL (integer, no decimals)
    uint256   entryGaugeProof;   // proof * 10 (e.g., 1200 = 120 proof)
    uint256   lastRegaugeRLA;    // regauged litres of alcohol, mL
    uint256   lastRegaugeABV;    // ABV * 100 (e.g., 5850 = 58.5%)
    uint256   lastRegaugeDate;   // unix timestamp (shows data freshness)
    uint256   estimatedVolume;   // model estimate, clearly separate from gauge
    CaskState state;
    string    warehouseId;
}

```

### Reserve Attestation (stays largely the same)

```

struct ReserveAttestation {
    uint256   physicalCaskCount;
    uint256   totalTokenSupply;
    uint256   reserveRatio;      // scaled 1e18
    uint256   timestamp;
    bytes32   attestationHash;
}

```

### Lifecycle Event (stays the same — event log)

```

event LifecycleTransition(
    uint256 indexed caskId,
    CaskState fromState,
    CaskState toState,
    uint256 timestamp,
    uint256 regaugeRLA,          // 0 if not a regauge event
    uint256 regaugeABV          // 0 if not a regauge event
)

```

### Optional: Reference Valuation (clearly labeled)

```

struct ReferenceValuation {
    uint256   caskId;
    uint256   estimatedValueUsd; // 1e18 scaled
    string    methodology;       // "age_curve_v1" | "market_comp_v1" | etc.
    uint256   timestamp;
    // Anyone can ignore this. The raw attributes are right there.
}

```

## 7. API REVISION

Endpoints reflecting actual data tiers:

Endpoint	Tier	Returns
GET /inventory	1	Cask count, IDs, existence attestation
GET /cask/{id}/attributes	1	Type, fill date, entry gauge, last regauge, state, warehouse
GET /cask/{id}/estimate	2	Current volume estimate, bottle yield estimate, model version
GET /cask/{id}/lifecycle	1	State history with timestamps and gauge data
GET /portfolio/summary	1+2	Aggregate counts, volumes, age distribution
GET /market-data	3	Auction comps, index values (reference only)
GET /cask/{id}/reference-valuation	3	Model-based valuation, clearly labeled as estimate

## 8. CRE WORKFLOW REVISION

### Workflow 1: Proof of Reserve

- Cron hourly
- HTTP → /inventory (1 call)
- EVM READ → totalMinted() (1 call)
- EVM WRITE → reserve attestation
- **Budget: 1 HTTP, 1 EVM read, 1 EVM write**

### Workflow 2: Physical Attribute Oracle

- Cron daily (attributes don't change fast)
- HTTP → /portfolio/summary + /cask/{id}/attributes for recently changed casks (2 calls)
- EVM WRITE → batch update cask attributes
- **Budget: 2 HTTP, 0 EVM reads, 1 EVM write**

### Workflow 3: Lifecycle Provenance

- HTTP trigger (webhook) for real-time events + Cron daily fallback
- HTTP → `/cask/{id}/lifecycle` on cron fallback (1 call)
- EVM WRITE → lifecycle event
- **Budget: 0-1 HTTP, 0 EVM reads, 1 EVM write**

All three well within CRE execution limits.

## 9. DEMO NARRATIVE (REVISED)

The pitch is no longer "we built a valuation oracle." It's:

*"Bonded warehouses maintain legally verified custody records — cask existence, type, age, gauge readings, lifecycle events. Today that data is locked in spreadsheets, emailed as PDFs, and arrives weeks late.*

*We built CRE infrastructure to pipe verified warehouse data onchain in real time. Proof of reserve. Physical attributes. Lifecycle provenance. Privacy-preserving.*

*We don't tell you what a cask is worth. We give you the verified facts to decide for yourself."*

## 10. WHAT WE BUILD FEB 6 (EXECUTION ORDER)

1. **Smart contract** (day 1-2): CaskAttributes struct, ReserveAttestation, LifecycleTransition events, onReport routing, read functions
2. **Valuation API** (day 2-3): FastAPI serving seeded cask data across all endpoints, clearly separated tiers
3. **CRE workflows** (day 3-5): PoR, attributes oracle, lifecycle. Get simulation working end-to-end.
4. **Privacy layer** (Feb 12+): Confidential HTTP swap on PoR inventory fetch
5. **Reference valuation model** (if time): Optional Tier 3 layer, clearly labeled
6. **Demo video** (Feb 25+): Record terminal demos, architecture diagram, narration