J

JCARUS
Blockchain Platform

Whitepaper

A gasless, USD-pegged evidence & metering protocol on Solana

Token: JCS · Standard: Token-2022 · Chain: Solana L1

Version 1.1 · 2026 · Technical edition

For information only. Not an offer of securities or financial advice.

Abstract

Jcarus is a protocol for anchoring tamper-evident records and metering usage on Solana without exposing end users to the friction of crypto. Enterprises reason entirely in US dollars; the protocol settles in its utility token, **JCS** (Token-2022, fixed supply of 1,000,000,000). Network fees are paid by the platform (gasless UX), operation prices are USD-pegged through a time-weighted average, and every enterprise draws from a prepaid on-chain JCS pool with on-chain overdraft protection. Each JCS consumed is atomically split four ways — burn, treasury, stakers, and the originating dApp — making JCS deflationary while aligning every participant in the network. This edition details the architecture, account model, cryptoeconomics, and threat model of the Phase-1 protocol, with system diagrams.

1. Introduction

Public blockchains provide verifiable, tamper-evident state, yet three barriers keep mainstream enterprises out. First, users must hold a volatile native asset to pay gas. Second, costs are denominated in that volatile asset rather than fiat, so a fixed business process has an unpredictable price. Third, key management and transaction mechanics are unfamiliar and operationally risky. Jcarus removes all three: the platform fronts network fees, quotes every operation in USD, and lets a business prepay a pool of JCS that its applications draw down as they anchor evidence or invoke integrated dApps.

1.1 Design goals

- **Predictable USD cost** for every on-chain operation, independent of token volatility.
- **Zero end-user gas** — the backend is the fee payer; users never hold SOL.
- **Authoritative on-chain settlement** — balances and payments are enforced by programs, not trusted services.
- **Extensibility** — multiple dApps and AI plugins meter against the same pool and treasury rails.
- **Least-privilege security** — the only online key can pay gas and spend within pool limits, nothing more.

1.2 Non-goals (Phase 1)

Phase 1 does not implement on-chain governance (parameters are multisig-controlled), an L2/rollup, or a native order book. These are deferred to later phases; the architecture is designed so they can be added without migrating token or pool state.

2. System Architecture

Jcarus separates three concerns. **Money movement** is on-chain and authoritative, implemented as five modular Anchor programs. **Orchestration and UX** live in a backend that is the gasless fee payer, prices operations, and indexes chain events into a fast read model. **Pricing** is derived off-chain from an oracle/TWAP and applied deterministically. The chain is always the source of truth for balances; the off-chain read model is reconciled by an event indexer and is never trusted for settlement.

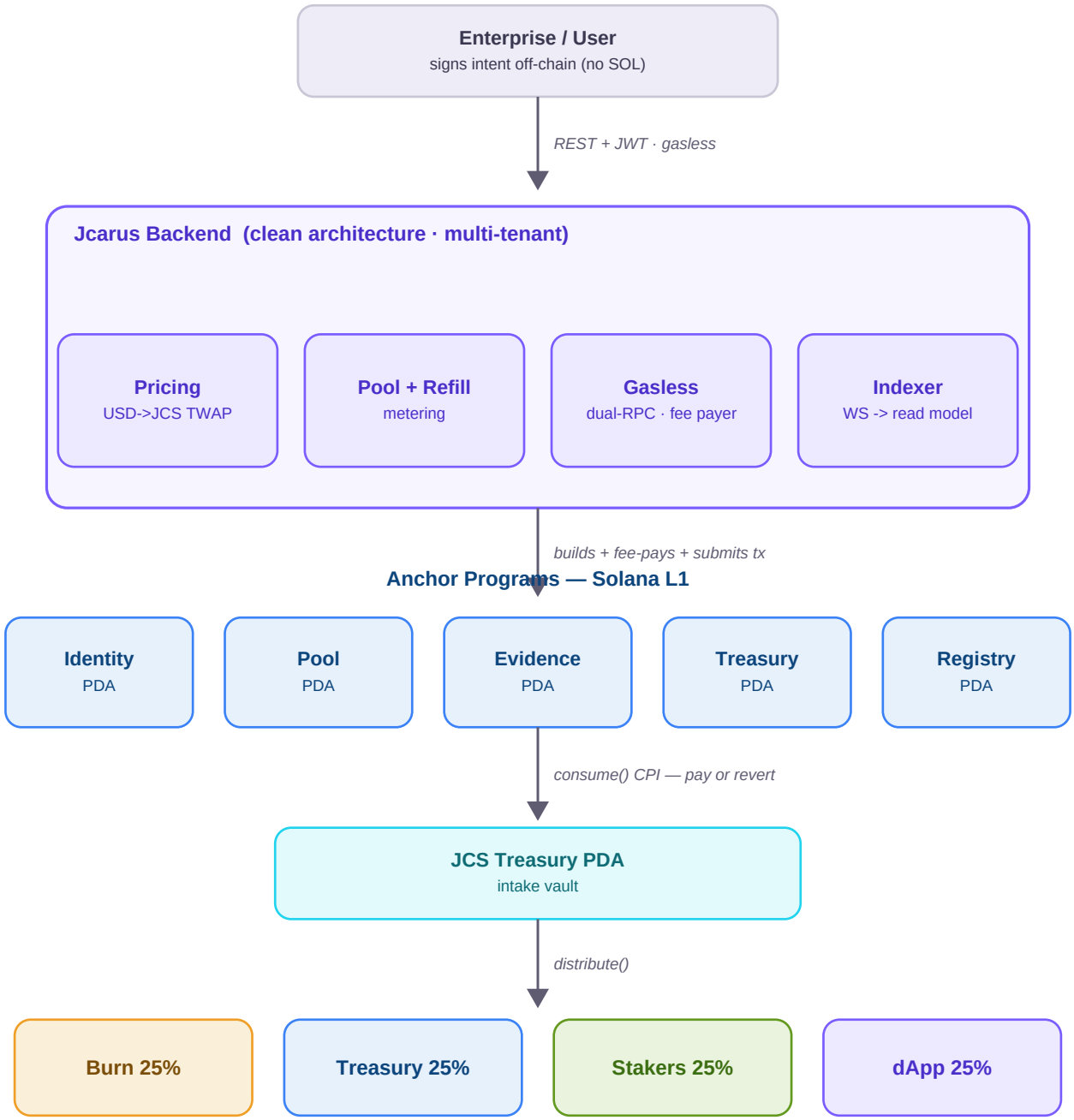


Figure 1 — System architecture: client intents flow through the backend, which prices, fee-pays and submits transactions to the on-chain programs; consumed JCS collects in the treasury and is split four ways.

2.1 Source of truth vs. read model

Every balance-changing effect is a program instruction. The backend mirrors results into PostgreSQL for fast API reads and billing, but all writes are idempotent projections of on-chain events (keyed on a monotonic nonce or transaction signature), so reconnects and replays never double-count and the read model always converges to chain state.

3. The JCS Token

JCS is the sole unit of protocol consumption, built on Solana's Token-2022 standard. Three extensions are enabled at mint creation and sized up front so the mint account never needs reallocation.

Property	Value
Standard	Token-2022 with TransferFee, MetadataPointer, Metadata
Decimals	9
Total supply	1,000,000,000 JCS (fixed, minted once)
Transfer fee	0.50% (cap 5 JCS/transfer); governance ceiling 1.00%
Freeze authority	Optional; off by default (regulated deployments only)
Inflation	None — mint authority revoked at launch

3.1 Fixed supply mechanics

The full supply is minted once to a distribution account, after which the mint authority is set to null. No protocol path can create JCS thereafter. Deflation comes from two sources: the burn share of every metered operation, and the transfer-fee mechanism, whose withheld fees are periodically harvested to the treasury. Metadata (name, symbol, URI) is written on-chain via the metadata extension, removing any dependency on an external metadata program.

4. Enterprise Pool Model

Each enterprise owns a program-derived pool holding a prepaid JCS balance in a vault whose authority is the pool PDA. Metered operations deduct JCS and forward it to the treasury intake.

4.1 Account and PDA layout

Account	Seeds	Key fields
Config	["pool-config"]	backend_authority, jcs_mint, treasury_vault
Pool	["pool", enterprise_id]	allocated_jcs, consumed_jcs, threshold_bps, nonce
Vault	["vault", pool]	token account, authority = pool PDA

4.2 On-chain guarantees

- **No overdraft** — available = allocated_jcs − consumed_jcs is checked before any mutation or transfer; a consume that would overdraw reverts (InsufficientPoolBalance).
- **No accounting drift** — JCS is moved with checked transfers, so consumed_jcs cannot diverge from the vault balance.
- **Threshold signalling** — when available ≤ allocated × threshold_bps ⁄ 10000 (default 20%), a RefillNeeded event fires exactly once per dip via a latch that clears on the next above-threshold refill.
- **Authorization** — only the configured backend authority or the pool manager may consume.

When auto-refill is enabled, the backend acquires JCS (TWAP-sliced, slippage-protected) and tops the pool back up, keeping service uninterrupted.

5. Pricing Engine (USD Peg)

Customers reason in USD; the protocol settles in JCS. The required amount for an operation priced at usd_price is:

required_jcs = usd_price / jcs_market_price(TWAP)

5.1 Smoothing and manipulation resistance

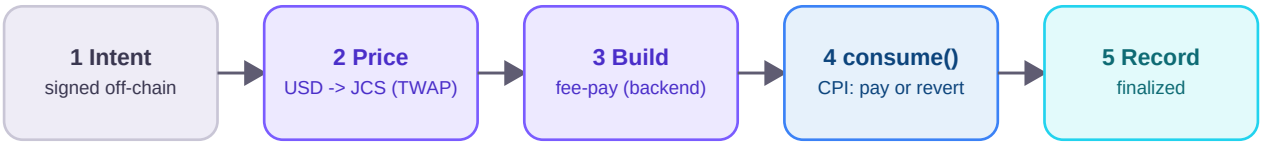
The market price is a time-weighted average over a configurable window (default 300 seconds) sourced from an oracle or DEX aggregator. A deviation guard rejects any spot sample that departs from the running TWAP beyond a configured bound (default 20%): such samples are discarded and the last good TWAP is served, protecting customers from transient spikes and the protocol from oracle manipulation. Quotes are cached for a short TTL to bound oracle load.

5.2 Worked example

With a TWAP of $0.50 per JCS, an operation priced at $0.50 requires 1.0 JCS; at a TWAP of $0.40 the same operation requires 1.25 JCS. The USD cost the customer sees is constant; the JCS quantity floats inversely with price.

6. Gasless Transactions

The enterprise signs an intent — a canonical, nonce-bound payload — off-chain, requiring no SOL. The backend verifies the signature against the enterprise's registered key and a strictly-increasing nonce *before* spending any SOL, so it cannot be induced to pay for unauthorized work. It then builds the transaction, pays the SOL fee, and co-signs as the pool's authorized consumer.



If step 4 fails (insufficient pool), the whole transaction reverts — no record is written.

Figure 2 — Transaction lifecycle: pricing and payment are enforced in the same atomic transaction as the record write.

6.1 Reliability

Submission uses a dual-RPC pool with round-robin plus failover and exponential backoff, and accepts only *finalized* confirmation. A bounded SOL float on the fee-payer wallet is monitored with low/high-water alerts and automated top-ups.

7. On-chain Programs

Five modular Anchor programs, each owning its state, exposing role-gated instructions, emitting events for the indexer, and (where it moves value) owning a PDA token vault.

Program	Key instructions	Holds funds
Identity	initialize, create_identity, assign_role	no
Enterprise Pool	create_pool, refill, consume, set_pool_config	yes
Evidence	init_counter, anchor_record, verify_leaf	no
Treasury	initialize, update_split, distribute	yes
dApp Registry	register_dapp, stake, unstake, set_revenue_share	yes

7.1 Atomic payment via CPI

Evidence anchoring invokes the pool's consume instruction first, by cross-program invocation, using the backend authority as signer. Payment and record-writing therefore occur in a single transaction: if the pool cannot pay, the consume reverts and the evidence record is never created — pay-or-nothing. Merkle roots let a single anchored record attest to a large off-chain batch; a supplied proof verifies any leaf against the anchored root.

8. Fee Distribution

Every JCS routed to the treasury is split on distribution. Shares are basis points that must always sum to 10,000 and are configurable via governance; floored shares leave rounding dust that rolls into the treasury reserve, so the intake vault always fully drains.

Share	Default	Purpose
Burn	25%	Deflationary sink
Treasury	25%	Protocol runway and operations
Stakers	25%	Secures and aligns dApp integrators
dApp owner	25%	Ecosystem incentive to the originating dApp

9. Wallet & Cross-chain Swap

A consumer wallet provides balance, receive, transfer, and swap. Because JCS lives on Solana, every swap routes in two hops through USDC: JCS→USDC on Solana via a DEX aggregator, then USDC→target across a bridge. Same-chain settlement (USDC) takes seconds; EVM targets route through fast bridges; native Bitcoin routes through cross-chain DEXs that deliver native BTC without wrapped tokens.

Target	Route	Rail (example)	Settlement
USDC	JCS -> USDC	Jupiter (Solana)	~seconds
BNB	JCS -> USDC -> BNB	Jupiter + deBridge	~30s
BTC	JCS -> USDC -> BTC	Jupiter + Chainflip	~10-15m

9.1 Custody

The recommended model is mobile-first with hardware-backed key storage and MPC by default: users onboard with email/passkey, keys are threshold-shared (no single point of compromise), and advanced users can export to self-custody. The gasless model is preserved throughout: the backend remains fee payer while the user's wallet signs transfer authority.

10. Tokenomics & Distribution

Supply is fixed at one billion JCS with no emissions. The illustrative allocation below balances community participation, ecosystem incentives, protocol runway, and long-term team alignment via vesting.

Allocation	Share	Notes
Community / sale	20%	Public + community rounds
Ecosystem & rewards	25%	Staking, grants, incentives
Protocol treasury	20%	Runway, operations
Team	15%	Long-term vesting
Liquidity	12%	DEX/CEX depth
Advisors	8%	Vesting

Illustrative allocation for this document; replace with official figures prior to any distribution.

11. Economic Security & Incentives

Value flows from real usage: enterprises spend USD, the buy module acquires JCS on the market, that JCS is consumed and split. The burn share creates continuous buy-and-burn pressure proportional to network activity; the staker share rewards dApps that lock JCS in the registry, which also provides Sybil resistance (a dApp must stake to be active and to earn revenue share); the treasury share funds runway; and the dApp-owner share incentivizes integrators to drive demand. Because supply is fixed, sustained usage is structurally deflationary.

12. Security & Threat Model

- **Overdraft & replay** — balance checks precede mutations; a monotonic per-pool nonce plus idempotent projections defeat replay and double-counting.
- **Value integrity** — all transfers are checked (mint + decimals), so counters cannot drift from vaults.
- **Key hierarchy** — program upgrade authority under multisig/timelock; mint/fee authorities under multisig; mint authority revoked at launch. The only online key (backend fee payer) can pay gas and consume within pool limits — never mint, never move treasury funds, never exceed a pool balance.
- **HSM-ready** — the fee-payer signer is an interface backed by a KMS/HSM in production; keys are rotatable.
- **Incident response** — a pause switch halts consume/refill without touching funds; a hash-chained, append-only audit log enables out-of-band tamper detection.

12.1 Threats and mitigations

Threat	Mitigation
Oracle manipulation	TWAP window + deviation guard reject outliers

Unauthorized spend	Intent signature + nonce verified before fee spend
Pool drain / overdraft	On-chain balance check reverts the transaction
RPC outage	Dual-RPC failover, finalized-only confirmation
Key compromise (online)	Least-privilege fee payer; HSM; rotation

An independent audit, dependency auditing, fuzzing of pool and pricing math, and a testnet soak with RPC/bridge failover chaos are prerequisites before mainnet.

13. Deployment & Operations

The backend runs as a container on a regional, auto-healing instance group behind a global HTTPS load balancer, backed by high-availability managed PostgreSQL with point-in-time recovery and private networking. Secrets are held in a managed secret store; the fee-payer key is backed by a KMS/HSM. A CI pipeline builds and audits programs and backend on every change; a tagged pipeline deploys via infrastructure-as-code with rolling updates. The indexer continuously reconciles the off-chain read model against chain state, with alerting on lag, refill failures, and pool-empty events.

14. Roadmap

Phase	Milestone
Q2 2026	Core protocol: 5 programs, JCS Token-2022, gasless backend, devnet demo.
Q3 2026	Multi-chain wallet, 2-hop swaps, community raise.
Q4 2026	Independent audit, multisig/timelock, mainnet, bug bounty.
Q1 2027	dApp registry & AI-plugin metering, staking rewards.
2027+	Governance DAO for fee/split parameters.

15. Disclaimer

This whitepaper is provided for information purposes only. It does not constitute an offer or solicitation to sell shares or securities, investment advice, or a recommendation of any kind. Forward-looking statements, token allocations, roadmap dates, and figures are illustrative and subject to change. Digital assets involve significant risk; participants should conduct their own research and consult professional advisors. Nothing herein is a promise of future performance or value.