KasForge

SECURITY REVIEW · PUBLIC EDITION

KasForge v1 Launchpad contracts

The contracts every KasForge coin launches and trades on, on Kaspa mainnet: what they enforce, who can do what, and how they were tested in Kaspa's own consensus engine.

RELEASE
KasForge v1 · contracts 3.8

NETWORK
Kaspa mainnet, live since 7 November 2026, 16:30 UTC

REPORT DATE
10 November 2026

PREPARED BY
PumpForge (self-run review)

SCOPE
9 contracts · transaction builders · service rules

RESULT
No Critical, High, Medium or Low issue in the release; 12 informational notes

1 Summary

KasForge is PumpForge's token launchpad built natively on Kaspa L1. Every coin is a standard KCC-20 token that trades on a bonding curve, graduates into a pool with permanently locked liquidity, and pays its creator a share of every trade. All of it is enforced by covenant contracts on Kaspa itself: there is no bridge, no sidechain and no custody, and every transaction is signed in the user's own wallet.

This report covers the contract release KasForge launched on mainnet with (KasForge v1, contracts 3.8). It describes what each contract enforces, what the keys KasForge runs can and can't do, the test campaign behind the release and the notes integrators and users should know. It is a self-run review published by PumpForge; the contracts have not been independently audited.

0	12	301	317
CRITICAL · HIGH · MEDIUM · LOW	INFORMATIONAL NOTES	TESTS PASSED ON THE RELEASE	RED-TEAM ATTACK TXS

Headline results.

- No tested transaction moved anyone else's KAS or tokens: not the market's, not a holder's, not an order owner's, not the creator's or the platform's fees.
- Every entry of every contract was run as real transactions in rusty-kaspa's consensus engine, honest and tampered.
- Every check line of every contract was removed one at a time to see that a test fails; each survivor was examined and is listed by contract.
- The keys KasForge runs can settle, sweep and pay only what the contracts allow; none can move a user's coins (section 4).

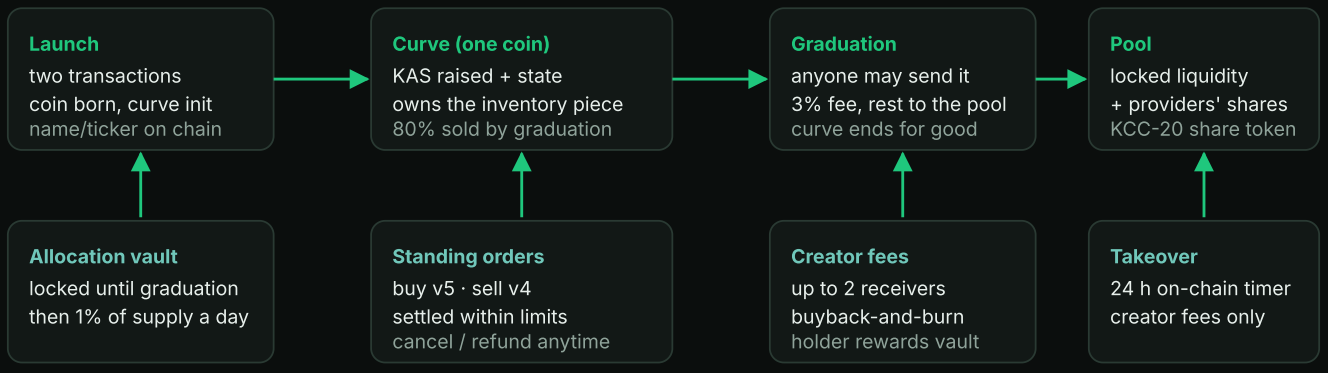
2 Scope

CONTRACT	VERSION	SOURCE	LINES	ROLE
ForgeCurve	3.8	`forge_curve.sil`	1295	The bonding curve: launch, buys, sells, fee claims, buyback-and-burn, holder payouts, fee settings, community takeover, graduation
ForgePool	3.8	`forge_pool.sil`	1151	The pool a coin graduates into: trades, liquidity and its share token, the same fee entries
ForgeBuyOrder	v5	`forge_buy_order.sil`	138	A standing buy, settled by the sequencer within the buyer's limit
ForgeSellOrder	v4	`forge_sell_order.sil`	130	A standing sell, settled within the seller's limit
Allocation vault	v4	`forge_vault.sil`	181	Holds a coin's free dev allocation until graduation, then releases 1% of the supply a day
ForgeTakeover	v1	`forge_takeover.sil`	27	The 24-hour timer of a community takeover
ForgeRewards	v1	`forge_rewards.sil`	34	Holds a coin's holder rewards until they are paid out

CONTRACT	VERSION	SOURCE	LINES	ROLE
ForgeHold	v3	`forge_hold.sil`	37	Holds a vested dev buy between the launch and its vesting
KCC-20 token	reference	`kcc20.sil`	270	Every KasForge coin and every pool share token; template hash `9703112e…`

Also in scope: the transaction builders and service rules of the KasForge engine (`forgekas`): quotes, order settlement, fee sweeps, holder-reward payouts, graduation, liquidity and the anti-snipe limits. The token locker scripts shared with ForgeVault are covered by ForgeVault's own published review. Compiler: SilverScript 1.0.0; consensus: rusty-kaspa at the revision SilverScript builds against.

3 How a KasForge coin works



Markets

A coin's curve is one coin on chain holding the KAS raised and the market's state; it owns one KCC-20 piece, the inventory, holding the tokens not yet sold. Every trade spends both and recreates them, and the curve's script checks every field of the next state, every output it makes and every KAS amount. When the raise is reached, anyone may send the graduation: the curve ends for good and a pool is born with 20% of the curve's tokens and the KAS raised minus a 3% graduation fee, at the curve's last price. The liquidity a coin graduates with can never be withdrawn; providers who add their own liquidity receive pool shares, a KCC-20 token of the pool's own.

Fees

A trade costs 1.5%: on the curve 1% platform and 0.5% creator, in the pool 0.75% platform, 0.5% creator and 0.25% to liquidity providers. Fees build up inside the market and leave only to the wallets the market records: the platform fee wallet (fixed per coin), the creator, up to two receiving wallets, and the coin's holder-rewards address. Buyback-and-burn spends a chosen share of creator fees on the coin and keeps the tokens locked in the market forever. Anyone may send a claim, a buyback or a holder payout; each pays only the recorded wallets.

Orders

A standing order holds the buyer's KAS (or the seller's tokens) under a script that says exactly what it accepts: which market, the least it must receive, and when it expires. The sequencer settles orders against the latest market coin without the owner's signature, and can only settle them within those limits. The owner can cancel at any time, and anyone can refund an expired order to its owner.

4 Keys and trust

KEY	CAN	CAN'T
Sequencer KasForge service	Settle standing orders within their limits · add tokens to a holder's balance when settling their buy · sign the partner share of a platform sweep (at most half, to the partner payout wallet) · pay holder-reward claims from a coin's rewards address · co-sign moving a vested dev buy into its vesting (with the dev) · cancel a pending takeover	Move, reduce or redirect anyone's tokens or KAS · settle an order outside its limit · change a market's state outside its entries · take any fee
Admin	Propose a community takeover of a coin's creator-fee settings (applies after a 24-hour on-chain timer) · move a coin's holder rewards if the service can't pay them	Touch tokens, the KAS raised, liquidity or platform fees · apply a takeover early · change anything but creator-fee settings
Platform fee wallet	Receive platform fees (fixed per coin at launch)	Anything else; it signs nothing in the contracts
Creator / fee authority	Claim creator fees (anyone may send the claim) · one fee-settings change that may only add to others' shares	Reduce receivers' shares · touch the market, its tokens or KAS
Anyone	Trade · send buybacks, claims, holder payouts, graduation, expired-order refunds, a takeover's finish after 24 h	Pay any of these to anyone but the recorded wallets

If the KasForge service is unavailable, nothing is lost: coins stay in their holders' wallets and the locked liquidity stays locked. The contracts live on Kaspa itself, so anyone can build transactions that follow their rules (the partner kit and SDK document every one) and send them through any public Kaspa node: trade directly on a curve or pool, settle standing orders (no signature needed), cancel their own orders and refund expired ones, send creator-fee claims, platform sweeps, buybacks and graduations (each pays only the recorded wallets), add or remove their own liquidity, collect unlocked tokens and release the daily dev allocation. Only four things wait for the service's key: holder-reward payouts (KF-01), the partner share of a platform sweep, adding tokens to an existing holder's piece (without it a buy makes a new piece), and cancelling a pending takeover.

5 Method

- **Real consensus.** Every builder's transactions are mined in blocks by rusty-kaspa's own consensus engine (Toccata rules on) and must be accepted exactly as built; tampered copies must be refused by the same engine.
- **One test per check.** Every check line of every contract was removed, one at a time, and the whole suite run against the weakened contract: a check that no test notices has no evidence behind it. Each survivor was examined and either covered by a new test or shown to be a second guard (another line, or the KCC-20 token's own rules, refuses the same transaction).
- **Model fuzzing.** Random sequences of buys, sells, orders, claims, buybacks, fee changes, graduation and liquidity moves, every step compared with an independent model of the market and checked against its invariants (the market always holds what it owes; tokens are conserved).
- **Quotes.** Random trades, each moving the price for the next, comparing the service's quote with what settles, to the sompi.
- **Bug classes from UTXO smart-contract audits.** The bug classes public audit reports found in UTXO-model (eUTXO) DEXes and launchpads on Cardano were each turned into transactions against KasForge (section 6.3).
- **Two-contract search.** Honest transactions were merged in pairs (two markets, a market and an order, two orders, a market and the allocation vault, a curve and a pool) and allowed to share outputs, looking

for one payment counted twice (double satisfaction).

- **Substitution lint.** A check reads every contract source and lists each place a transaction builder chooses where a payment goes, and every token input not tied to the market's own token. The list is pinned: a change that adds an unchecked position fails the suite.
- **Market state model.** An attacker searches combinations of transactions over curve, pool, orders and vault (stray coins and pieces, fake tokens, reused outputs, two markets at once), checking safety (nobody takes what isn't theirs) and liveness (refunds and claims always remain possible).
- **Auditor-view checks.** What each service key can and can't sign; dust and storage mass of every action; fee and price economics (including trading around a known order); every reported case replayed as a real transaction.
- **Published audits as checklists.** Public audit reports of UTXO-model DEXes and launchpads (Cardano: SundaeSwap by Runtime Verification, SundaeSwap V3, Minswap V2, Genius Yield, Splash, Jam on Bread; Trail of Bits' work on CAT; Bitcoin Cash: the Cauldron DEX in CashScript; Chia) were read and each relevant bug class turned into tests (section 6.3).
- **Red-team tool.** The open-source `kaspa-redteam` tool (kasperolabs) attacked honest transactions of every entry from an outsider's seat and from a user's seat (section 6.4).
- **Separate source reviews.** Over the review cycle the contracts went through several separate source reviews by three different reviewers, each given the sources, a design brief and the known trade-offs and asked to break them; every reported case was rebuilt as a real transaction and either shown to be refused or fixed before release. The final release's sources were reviewed again, with property campaigns over fee settlement (700,000 configurations), the anti-snipe rules (a 640-case matrix) and fee changes (one million cases); every reported case was rebuilt as a real transaction. They reported no issue in the release.

6 Results

6.1 Test suite

AREA	WHAT IT COVERS
`engine`	Every builder, every contract entry, the checks written against each contract line, quotes against settlement
`consensus`	Full coin lives in rusty-kaspa's consensus engine: launch, trades, orders, claims, graduation, liquidity
`consensus_markets`	Markets in real consensus: many coins, orders and claims at once
`fuzz_math`	Random trade sequences checked step by step against an independent model
`market_model`	The market maths against a reference model
`mutations`	Tampered transactions that must all be refused
`order_templates`	Order contract templates pinned to the published hashes
`partner_kit`	The partner kit's transactions accepted exactly as documented
`pieces`	One balance per coin: pieces, top-ups and sends
`presign_fixtures`	Pre-signed transaction fixtures
`review_curve`	Curve review cases rebuilt as transactions
`review_dust`	Dust, deposits and storage mass
`review_economics`	Fee and price economics
`review_keys`	What each service key can and can't sign

AREA	WHAT IT COVERS
`review_lint`	Source rules every contract must follow
`review_pairs`	Contract pairs in one transaction (double satisfaction)
`review_repro`	Reported cases replayed as real transactions
`sdk_vectors`	SDK test vectors
`streams`	Payment streams
`v3`	Launch, allocation, vesting and one-balance holdings
`v31`	Fee settings and creator-fee sharing
`vault_consensus`	Locks and vesting in real consensus
`vault_model`	Lock and vesting maths against a model
`wallet`	Signing through the same path browser wallets use
`wallet_units`	Wallet signing units and limits

301 tests passed, 0 failed on the release before it went to mainnet, plus the multi-hour fuzz and deletion campaigns below.

6.2 One test per check (deletion check)

CONTRACT	LINES A TEST CATCHES / CHECK LINES
ForgeCurve	242 / 374
ForgePool	249 / 409
ForgeBuyOrder	12 / 19
ForgeSellOrder	13 / 23
Allocation vault	25 / 56
ForgeTakeover	1 / 1
ForgeRewards	5 / 8
ForgeHold	8 / 9

Every surviving line was examined. They fall into: second guards (another line refuses the same transaction), the KCC-20 token's own balance and borrow rules, checks on transactions only the signer's own coins pay for, overflow bounds that today's parameters can't reach, and checks that only matter for hand-built launches the service would refuse anyway. Every line changed in the final release was checked the same way; all were caught except one unreachable branch (a fee change on a coin with holder rewards owed but no receiving script, which no transaction can create).

6.3 UTXO bug classes

CLASS	WHAT GOES WRONG	HOW KASFORGE HANDLES IT (TESTED)
Double satisfaction	Two contracts in one transaction satisfied by one output	Payouts are tied to each market's own continuation (its own output + 1 and after), and a script input can't sit at a payout index. Every market entry tested with a second market, an order or a stray piece in the same transaction.

CLASS	WHAT GOES WRONG	HOW KASFORGE HANDLES IT (TESTED)
Missing input authentication	Spending an input the contract doesn't check	Every entry that doesn't trade refuses the market's tokens and the pool's share tokens as inputs; the market's own pieces are read and checked (owner, owner type, borrow setting, amount, ≥ 1 KAS).
Foreign or stray tokens	Tokens sent to a market from outside	A market only uses pieces of its own coin owned by itself, with the exact amount its state records. Stray pieces can't stand in for its inventory profitably.
Unchecked outputs	A value or script left free	Every output the market makes is checked: its script (template and state), its covenant, its KAS. The next state is pinned field by field in every entry.
Caller-chosen positions	Indexes passed as arguments	Every index argument is checked against what it must point at; tampering with any of them is refused (tested).
Time handling	Locks and deadlines	Ages come from the block that confirmed a coin (takeover, order expiry, anti-snipe); lock times only bound from below.
Locked or lost value	Dust and deposits	Deposits return to holders; payouts below the minimum wait instead of failing; the largest transaction (graduation with an allocation, burned tokens and a pending takeover) has mass 150,364, under the 500,000 block limit.
Signature scope	Signatures that don't commit to outputs	Every signature in a contract entry must be SIGHASH_ALL (launch, fee changes, takeovers, partner shares, rewards, holds).
Arithmetic	Overflow and rounding	Bounds are checked on chain (supply × curve size, pool product), and rounding favours the market (KF-09).

6.4 Red-team tool

`kaspa-redteam` 0.16.0 (attack library 0.9.1), judged by the rusty-kaspa consensus validator in process. A transaction counts as a finding only when consensus **accepts** an attack the tool expected to be refused. Three runs: a source search over every contract (0 findings, 70 attack transactions); 42 honest transactions covering every launchpad entry, attacked from an outsider's seat (317 attack transactions); and the same transactions from a user's seat (buyer, order owner, liquidity provider), asking whether a user can take more than the contract allows.

ATTACK	SHAPE	RESULT (OUTSIDER SEAT)
`OUT-01`	Correct value, attacker's script	refused (all tries)
`OUT-03`	Valid first output, skim a later one	refused
`FEE-01`	Route value to the miner fee	refused
`IN-01`	Two covenant coins sharing one set of outputs	refused
`IN-02`	Attacker's coin first, covenant moved behind it	refused
`OUT-02`	Correct script, reduced value	flagged only where the sender trims a deposit it pays itself (KF-06)
`OUT-04`	Keep what fixed payouts leave over	flagged only for the ≤ 0.01 KAS network-fee budget of whoever sends a refund, graduation or release

No attack moved anyone else's KAS or tokens. From the user's seat the only accepted changes were a buyer trimming the deposit on their own new piece and an order owner cancelling their own order to themselves.

6.5 Limits that hold

PROPERTY	RESULT
Largest transaction (graduation with an allocation, burned tokens and a pending takeover)	mass 150,364 · block limit 500,000
Quotes against settlement (24 random trades, each moving the price)	equal to the sompi
Model fuzz (150 random seeds of full coin lives)	every step matches the model; no invariant broken
Maths fuzz (70 runs, about 25,000 transactions verified like a node)	no tokens or KAS appeared or vanished; every quote matched; the pool never lost value per share; buying and selling straight back never made money
Sequencer under junk coins (many tiny coins sent to its address)	settles still fit a block: it pays with its largest coins and merges small ones when idle
Market tokens	supply conserved; burned tokens stay in the market for good

6.6 Attack catalogue

Every attack below was built as a transaction against the release's contracts and run through Kaspa's script engine or full consensus. All are refused.

AREA	ATTACK TRIED	RESULT
Fees	Two coins' platform sweeps (or a curve and a pool) in one transaction, pointing at one shared payment	refused
Fees	Two coins' creator claims of the same creator sharing one payout	refused
Fees	A creator claim pointing its payout at the output an order pays its owner (order settle or refund in the same transaction)	refused
Fees	A graduation fee and another coin's sweep sharing one payment	refused
Fees	Sending a claim (anyone may) and keeping a receiving wallet's part	refused
Fees	Moving, cutting or turning a holder-rewards receiver into a key, by a fee change or a claim	refused
Fees	A fee change signed SIGHASH_NONE, or by the wrong key, or a second fee change	refused
Fees	A fee change setting buyback or receivers' shares above 100%, or a negative split	refused
Fees	A platform sweep with a partner share not signed (SIGHASH_ALL) by the sequencer, or a partner share above half	refused
Fees	A trader making itself the fee authority or a receiver, or planting a takeover timer, inside an ordinary buy	refused
Trades	A buy minting the buyer's piece to an outsider, or handing the market's remaining inventory to the buyer's own key or covenant	refused
Trades	A buy or sell recording any different next state (reserve, KAS raised, fees owed, flags, vault, shares, burned, claimable)	refused
Trades	A direct sell taking more KAS out of the market than the price	refused
Trades	Wrong inventory or output positions passed as arguments	refused
Trades	A stand-in token piece (a stray piece owned by the market) used instead of the inventory	refused
Trades	The market's inventory or the pool's unissued share tokens taken along by a claim, buyback, fee change or takeover entry	refused

AREA	ATTACK TRIED	RESULT
Trades	A zero buy, an extra mint, a sell that leaves the market short	refused
Anti-snipe	More than 1% of the supply in one buy in the first minute, including chained unconfirmed buys and top-ups	refused
Orders	Settling a buy order below its slippage limit (fresh piece and top-up)	refused
Orders	The settler taking the bought tokens, naming its own coin as the payment, or keeping the 1 KAS token deposit	refused
Orders	Selling part of a sell order's tokens and keeping the rest	refused
Orders	Refunding an expired sell order with a stray token instead of its own tokens, or to anyone but the owner	refused
Orders	Settling an order against a claim, a fee payout or another market	refused
Orders	Settling an order against a market coin made after its expiry	refused
Launch	A launch minting more dev tokens than it paid for, a free dev piece without a dev buy, an inventory bigger than the supply or minted to a key	refused
Launch	A free allocation minted bigger, or straight to the launcher instead of the vault, or a vault born already started	refused
Launch	A launch signed SIGHASH_NONE, or someone else initialising a launcher's curve	refused
Launch	A curve born with a hidden sibling coin of its own covenant	refused
Graduation	Graduating before the raise is complete, or keeping the curve alive next to its new pool	refused
Graduation	A new pool with any wrong field (token, reserve, KAS, fees owed, settings, shares), a pool or token coin holding less, or the pool's tokens sent to a key	refused
Graduation	A sibling coin in the new pool's covenant	refused
Graduation	Pointing the allocation vault at a counterfeit token, or starting its clock outside graduation	refused
Allocation vault	A release paying anyone but the beneficiary, more than has unlocked, or rewriting who later releases pay	refused
Allocation vault	Skimming the vault coin's KAS or a topped-up piece's KAS	refused
Takeover	Finishing before 24 hours, finishing with no proposal (a look-alike timer coin), or with another coin's timer	refused
Takeover	Destroying, refreshing or re-creating a pending takeover's timer inside any other entry	refused
Takeover	A proposal dropping the holder-rewards receivers, or a finish leaving a timer behind or not paying the timer's KAS back	refused
Takeover	A takeover finish rewriting the market's other state	refused
Liquidity	A share token made beforehand, made twice, owned by the binder, or with extra shares for a key	refused
Liquidity	A provider taking the pool's token or share inventory, or making either borrowable, in a deposit or withdrawal	refused
Liquidity	Withdrawing the liquidity the coin graduated with	refused
Holder rewards	A payout signed SIGHASH_NONE or by any key but the service's; a recover by any key but the admin's	refused
Dev-buy hold	Taking held tokens before the late date, or sealing with only one of the two signatures	refused

6.7 Service and site

CHECK	WHAT IT GUARANTEES
Pre-sign check	Every build states the total KAS it takes from the wallet; the site recomputes that from the transaction itself and refuses to sign if it is missing or higher.
Server copy	The service attaches the wallet's signatures to its own copy of the transaction, so nothing can change between signing and sending.
API inputs	Every number field is range-checked before any price maths runs.
Sequencer funding	Settles pay with the sequencer's largest coins; junk coins can't make a settle too big for a block.
Records	The service follows each market on chain: a market moved by a transaction KasForge didn't send is re-read on the next round, so outside trades show within seconds.
Anti-snipe	The 1%-per-wallet total for the first minute is applied by the service on top of the on-chain 1%-per-buy rule (KF-10).

7 Notes on the release

ID	SEVERITY	STATUS	TITLE
KF-01	INFORMATIONAL	BY DESIGN	Holder-reward payouts rely on the KasForge service
KF-02	INFORMATIONAL	BY DESIGN	Community takeover: the admin key can hand a coin's creator-fee settings to new wallets
KF-03	INFORMATIONAL	BY DESIGN	The sequencer may add tokens to a holder's balance
KF-04	INFORMATIONAL	ACKNOWLEDGED	One market coin per coin: busy coins queue
KF-05	INFORMATIONAL	BY DESIGN	Order expiry and refunds
KF-06	INFORMATIONAL	ACKNOWLEDGED	Whoever sends a transaction chooses the deposits it pays from its own coins
KF-07	INFORMATIONAL	ACKNOWLEDGED	A launch is two transactions
KF-08	INFORMATIONAL	DOCUMENTED	Integrators should verify a coin's lineage
KF-09	INFORMATIONAL	BY DESIGN	Rounding always favours the market
KF-10	INFORMATIONAL	BY DESIGN	The first-minute limit per wallet is enforced by the service
KF-11	INFORMATIONAL	BY DESIGN	A payout index may point at any equivalent output
KF-12	INFORMATIONAL	DOCUMENTED	Network-level attacks are outside this review

KF-01 Holder-reward payouts rely on the KasForge service

INFORMATIONAL BY DESIGN forge_rewards.sil · pay, recover

The share of creator fees a coin sends to its holders is paid by the coin's market into that coin's own rewards address. Who is owed what (pro rata by the tokens each holder's own wallet holds) is kept by the KasForge service, which pays each claim out of that address with its key. The signature is SIGHASH_ALL, so it commits to every output: a claim can only pay what the service built.

The platform's admin key can move a coin's rewards if the service can't (a lost record, a retired coin). The markets themselves never depend on this: their creator fees, liquidity and platform fees are enforced on chain.

KF-02 Community takeover: the admin key can hand a coin's creator-fee settings to new wallets

INFORMATIONAL BY DESIGN forge_curve.sil / forge_pool.sil · proposeTakeover, finishTakeover, cancelTakeover

If a coin's community asks (for example after its creator leaves), the admin key may propose new creator-fee settings. The proposal is a separate on-chain coin whose age comes from the block that confirmed it; it can only be applied once it is **at least 24 hours old**, and it can be cancelled until then.

A takeover changes only who receives the creator fees (and the fee authority). It cannot touch tokens, the KAS raised, the pool's liquidity, the platform fees or anyone's balance, and the proposal must keep any holder-rewards receivers the coin already has. While a proposal is pending, creator fees keep building up for the new settings.

KF-03 The sequencer may add tokens to a holder's balance

INFORMATIONAL BY DESIGN KCC-20 borrow guard · buy order settle

So that every wallet holds one balance per coin, pieces bought through KasForge carry a borrow guard: the sequencer key may add tokens to them when it settles that wallet's next buy. The token's own rules only ever let a borrowed piece grow, keep its owner and keep its KAS. The sequencer can never move, reduce or redirect a holder's tokens.

The only thing the guard allows beyond that is spending the piece's outpoint while adding to it, so a wallet's piece can change outpoint between two of its own transactions; the wallet then rebuilds. Holders who prefer no guard can move their tokens to a plain piece at any time.

KF-04 One market coin per coin: busy coins queue

INFORMATIONAL ACKNOWLEDGED design

Each curve or pool is one coin on chain, and every trade spends it. Standing orders let many buyers and sellers queue: the sequencer settles them one after another against the latest market coin, each within its own limit. A trade sent directly (by a partner or a router) spends one exact market coin; if another trade moved the market first, it is refused and rebuilt.

The same property stops sandwich attacks: a signed transaction spends one exact coin, so a front-runner can make it fail but never make it execute at a worse price.

KF-05 Order expiry and refunds

INFORMATIONAL · BY DESIGN · forge_buy_order.sil / forge_sell_order.sil · settle, refund

Kaspa scripts can only bound the current time from below. An order therefore refuses to settle against a market coin made after its expiry (the coin's own confirmation time, which nobody can fake), and after the expiry anyone may send its refund. A refund always pays the owner exactly what the order holds, minus a 0.01 KAS refund fee that covers whoever sends it. The owner can cancel at any time with their own signature.

KF-06 Whoever sends a transaction chooses the deposits it pays from its own coins

INFORMATIONAL · ACKNOWLEDGED · deposits · token pieces

Every new token piece carries a KAS deposit (about 1 KAS), returned to its holder when the tokens are sold. Where the sender of a transaction pays a new piece's deposit from its own coins, it may choose a smaller one. The contracts check every amount that belongs to someone else (the market's KAS and tokens, the owner's tokens, order payouts, the market's own pieces, which must carry at least 1 KAS); they don't force a sender to spend more of its own KAS than it chooses to.

KF-07 A launch is two transactions

INFORMATIONAL · ACKNOWLEDGED · forge_curve.sil · init

A coin's id only exists once its first transaction is built, so a launch is two transactions, approved one after the other. If a creator stops after the first, the uninitialised curve's 1 KAS deposit stays locked with it. Nothing else is at risk: no token exists until the second transaction.

KF-08 Integrators should verify a coin's lineage

INFORMATIONAL · DOCUMENTED · forge_curve.sil · init (covenant-id proof)

On Kaspa a covenant's birth runs no script. A KasForge curve proves its own birth in `init`: it recomputes its covenant id from the genesis outpoint, so a curve that was born with a sibling coin, or not born alone, can never start. Integrators listing KasForge coins from the chain should still compare a coin's script with KasForge's published templates (the service exposes them at `/api/v1/contracts`) or list only coins they saw launch.

KF-09 Rounding always favours the market

INFORMATIONAL · BY DESIGN · forge_curve.sil / forge_pool.sil · prices, fees, shares

Every amount a market pays out (tokens bought, KAS from a sale, fee splits, liquidity shares) rounds down, and what it takes in rounds up. The residue (at most a few sompi a trade) stays in the market. The fuzz campaign compares every step with an independent model of the market.

KF-10 The first-minute limit per wallet is enforced by the service

INFORMATIONAL BY DESIGN anti-snipe

In a coin's first minute one buy can take at most 1% of the supply, and the curve enforces this on chain. The 1%-per-wallet total in that minute is enforced by the KasForge service, because a script can't see a wallet's other transactions. A larger buy is not refused: it waits as an order until the minute ends. The creator is exempt (the dev buy).

KF-11 A payout index may point at any equivalent output

INFORMATIONAL BY DESIGN forge_curve.sil / forge_pool.sil · sell

Some entries take the index of the output that pays a key as an argument. It must point at an output to the right key holding at least the amount due, so it can point at an equivalent output (the payee's own change, for example). The outputs are fixed by the signer either way, so nothing moves differently, and Kaspa transaction ids don't cover signature scripts.

KF-12 Network-level attacks are outside this review

INFORMATIONAL DOCUMENTED scope

This review covers the contracts, the transactions KasForge builds and the service's rules. Mining and mempool behaviour (for example a miner delaying a transaction) was not tested; the contracts never rely on transaction ordering for safety, only for which of two competing trades settles first.

8 For integrators and users

- **Check the script.** Compare a coin's curve or pool script with the templates KasForge publishes (`/api/v1/contracts`) before listing it (KF-08).
- **Build exactly.** Every market entry pins its next state and outputs; the partner kit and SDK build them byte for byte. A trade that loses a race is refused, never filled at a worse price; rebuild it against the new market coin.
- **Orders are the safe default** for users: they can only fill within their limit, can be cancelled at any time and are refunded after they expire.
- **Deposits.** New token pieces carry about 1 KAS, returned when the tokens are sold.

9 What this review is

This is a self-run review published by PumpForge. The contracts have not been independently audited. Tests show the presence of the properties they check, not the absence of every issue; mining and mempool behaviour was not tested (KF-12). The contracts are immutable: every coin keeps the contracts it launched with, so this report stays valid for every coin launched on KasForge v1.

Docs: kaspa.pumpforge.tech/docs · Partners: kaspa.pumpforge.tech/partners/docs