What the Counted Money Is

Seven investigations of the agent payment ledger on Base: own positions, loops, the price of a rank, the time from mint to first payment, two populations that barely meet, the recipients we could not name, and a theft replayed against our own detector

Agentic Finance Graph — Research Note 02, version 1.0 · N. Simonyan, Agentic Finance Graph · Figures as of **2026-09-27 14:52 UTC**, read from the ledger's tables at one moment · CC BY 4.0 · agenticfinancegraph.com/research

Abstract. Research Note 01 defined the counted set: stablecoin payments from a wallet bound to an ERC-8004 identity, after the identity existed, that were not routing hops. A total built that way is honest about where the money came from; it says nothing yet about what the money did. We asked seven questions of it. *Own positions:* reading the receipt of every payment to a lending market or vault, $6.02M of $9.50M counted (63.4%) went into lending and vault positions credited back to the paying wallet itself; most of it was moved by 82 Safe smart accounts that never submit their own transactions. *Loops:* no two wallets in the counted set paid each other in both directions. *The price of a rank:* reaching L7 costs cents, and 60 ranked identities rest on less than a dollar, including 23 minted in one week that each paid one address $0.15; the cheapest L9 has spent $1,084. *Survival:* an identity minted on its own is 11 times as likely to make a payment within a year as one minted in a batch of a thousand or more. *Two populations:* ERC-8004 wallets paid 4.8% of the dollars that reached x402 Bazaar sellers on Base in the week measured. *Names:* 16 more recipients named from verified contract source took labelled coverage from 88.0% to 92.3% of counted value; one contract called OpenRouter turned out to be a swap router. *A theft replayed:* our own drain detector would have missed the May 2026 Bankrbot theft in three independent ways. Three corrections to our own record follow from the work, and every measurement is now a live metric with a published definition.

1. Why take the total apart

A counted total is a sum of decisions. Note 01 made the decisions that decide *whose* money a payment is: the binding, the mint date, the receipt test for hops. Those decisions are necessary and they are not sufficient. A reader who sees $9.50M of "counted agent payments" will read it as spending, as an economy of machines buying things, and every rank built on the same rows will be read as evidence of that economy. Before that reading hardens, we owed ourselves the checks a sceptical reader would run.

We ran seven, in the order such a reader would ask: How much of this money is still the agents' own? Do agents pay each other in circles? What does a rank cost to fake? How long after a mint does an identity pay, if ever? Are the identity registry and the payment rail the same population? Who receives the money we still cannot name? And would our own drain detector have caught the one theft we cite? Each answer below comes from our own tables, the chain and transaction

receipts, read on 27 September 2026. Each became a live metric with a definition. Three of them corrected things we had said in public, and Section 9 lists those corrections.

2. Own positions: how much of the "spending" is the agents' own cash

Note 01 already reported that the largest recipient of counted money is a contract whose verified name is *Adapter*, and that its documentation describes deposits into Aave and ERC-4626 vaults. A description is not a measurement. The question that matters for the total is transaction by transaction: when an agent sent stablecoins to a lending market, a vault, or an adapter in front of one, did the position come back to the agent? If it did, the money did not leave the agent's hands. It was moved from a checking account to a savings account.

2.1 The test

A deposit that credits the depositor leaves a trace in the receipt of the same transaction: the depositor is minted a claim token (vault shares, aTokens), or, in older Compound-style markets, the market sends the depositor its own token. For a counted payment t from wallet w_t to recipient c_t in transaction h_t, we define

$$\begin{aligned}\mathrm{own}(t) \iff \exists\, \ell \in \mathrm{logs}(h_t) :\ & \mathrm{Transfer}(\ell) \wedge \mathrm{to}(\ell) = w_t \\ & \wedge \big(\mathrm{from}(\ell) = \mathbf{0} \ \vee\ (\mathrm{from}(\ell) = c_t \wedge \mathrm{token}(\ell) = c_t)\big)\end{aligned}$$

and apply it to counted payments whose recipient is labelled as a lending market, a yield vault or a policy adapter in front of one. We restrict it to those categories on purpose. A swap also sends the payer a token in the same transaction, but a token bought is an exchange, not a position in a market, and a primary token sale can mint to the buyer; neither should be read as the buyer keeping its money. The counted value then splits exactly into two parts:

$$V(C) \;=\; \underbrace{\sum_{t \in C,\ \mathrm{own}(t)} v(t)}_{\text{own positions}} \;+\; \underbrace{\sum_{t \in C,\ \neg\,\mathrm{own}(t)} v(t)}_{\text{left the wallet's control on Base}}$$

2.2 What the receipts say

We read 1,371 receipts. 1,335 of them (97.4%) show the payer credited in the same transaction, for $6,017,368. That is **63.4%** of all counted value. What left the paying wallets' control on Base is $3,480,064, and even that includes bridges, where the money may still belong to the same owner on another chain.

Table 1. Counted payments to lending and vault recipients, by recipient, and how many the receipt proves were credited back to the payer.

Recipient	Payments	USD	Proven own	Proven USD
Adapter (Aave v3 / ERC-4626 policy adapter) `0xaffd…2c31`	876	$5,606,494	876	$5,606,494
Morpho: General Adapter1 `0xb98c…746a`	82	$145,895	77	$144,288
UsdcVaultL2 (USDC vault) `0x3128…e858`	11	$114,527	11	$114,527
Euler: EVault `0x4c1a…f61e`	23	$57,630	22	$57,607
Aave v3 aToken (ATokenInstance) `0x4e65…c0ab`	46	$30,351	23	$23,249
AmpleEarn `0x1688…9c0f`	293	$23,211	293	$23,211
Compound v3 Comet (CometWithExtendedAssetList) `0xb125…eb2f`	7	$12,425	7	$12,425
VaultV2InKind `0xc777…263e`	6	$11,880	6	$11,880
Fluid Liquidity (FluidLiquidityProxy) `0x52aa…e497`	4	$10,617	4	$10,617
RiseVault (vault) `0x0109…84fd`	6	$10,158	0	$0
Wasabi vault (WasabiVaultRecoveryV1) `0x1c4a…8b24`	6	$7,599	6	$7,599

The Adapter alone accounts for $5.61M in 876 payments, and every one of them is proven. Its verified source leaves no room for doubt about why: the implementation's notice reads "Wraps Aave v3 supply/withdraw and any ERC-4626 vault deposit/withdraw (Morpho, Fluid…)", every public function is named for the caller (`supplyForSelf`, `withdrawForSelf`, `depositVaultForSelf`, `redeemVaultForSelf`, and the asynchronous redeem pair), and a developer comment calls it "double-enforced recipient binding: this wrapper hardcodes `receiver = msg.sender`". The contract cannot deposit on anyone else's behalf. The few unproven rows elsewhere are deposits whose receipt credits a different address, or vaults that settle asynchronously; they stay on the "left" side until a receipt says otherwise.

2.3 Who the depositors are

This is where we nearly got it wrong. Our wallet table recorded all 82 of the Adapter's depositors as ordinary accounts, and a first pass read their transaction counts as "each wallet has signed exactly one transaction". The code said otherwise. Every one of the 82 addresses carries the same 171 bytes of contract code; storage slot 0 points to the Safe v1.4.1 singleton (Safe or SafeL2), for 82 of the 82 we read. They are Safe smart accounts, and a contract account's transaction count starts at one and stays there, so the "one signature" reading was an artefact of a stale column (Section 9). The account we opened has one owner and the *Safe7579* module adapter installed, the plumbing that lets an ERC-7579 session-key module act for the account without the owner signing each move.

So who moves the money? In a sample of 110 of the Adapter's transactions, 0 were submitted by the paying account. 70 came from one externally owned address, `0x4656…a28e`, which had sent 47,725 transactions by the day we looked, each calling `executeSinglechainOps` on a single entry

contract `0x0000…daaf`; 36 came through the ERC-4337 EntryPoint, submitted by 10 different bundlers. A second sample of 60, drawn differently by the live worker, found 0 submitted by the paying account and 19 by the same relayer. The money also comes back the same way: the largest senders of stablecoins *to* these accounts are VaultV2, MetaMorphoV1_1, MetaMorpho, VaultV2 contracts, which is what withdrawals from Morpho and similar vaults look like.

It is concentrated. The largest depositor is 41.9% of the Adapter's volume and the top five are 83.1%; the flow began on 2026-05-19. Every one of the 82 identities was minted by its own account (82 distinct minters), which is why Note 01 described them as independent. Their behaviour says they are a fleet: identical account code, the same module adapter, the same relayer, the same contract, their own money going in and coming back out.

What we do not assert. We do not know who operates the fleet, and a relayer can serve many operators; we name no company. We do not claim the positions are still held today. And the definition of the counted total does not change: a deposit is still a stablecoin transfer out of a bound wallet. What changes is how we publish it. The total now travels with its split, the new metrics `observed_pay_usd_own_positions` and `observed_pay_usd_left_control` (definitions `pay.usd.ownpositions.v1`, `pay.usd.leftcontrol.v1`), and our house rule gains a clause: quote the counted total beside the largest wallet's share, the median payment, and the part that went into the payers' own positions.

3. Loops: do agents pay each other in circles?

A total can be inflated without any money leaving a circle: A pays B, B pays A, and both payments are "spending". We measured the reciprocal value over all pairs of senders and recipients in the counted set,

$$R \;=\; \sum_{\{a,b\}} \min\big(V(a\to b),\; V(b\to a)\big),$$

and the three-step cycles $a \to b \to c \to a$. Both are zero: **0** reciprocal pairs and **0** three-step cycles. Payments from one ERC-8004 wallet to another are small, $28,404 in 328 payments from 142 payers, 0.30% of the total.

A loop can also run through a venue: pay a market, get the money back later. We can only test that where we record inflows, which today is 368 of the 1,423 ranked agents. For them, $1,075,535 went to 941 counterparties and $210,900 (19.6%) came back from the same address within thirty days of the last payment; 256 pairs got at least half of it back. That is mostly Section 2 seen from the other side, deposits and withdrawals at the same market, and it is why we measure own positions from receipts rather than from round trips. We do not publish the venue figure as a metric until inflow coverage matches payment coverage.

4. The price of a rank

The liveness ladder rewards behaviour that costs something. L7 needs a counted payment to a different address after the identity existed; L8 adds a stablecoin float of at least $10; L9 adds three distinct counterparties in thirty days. On Base the costs are small. In the cheapest case,

$$c_{L7} \approx g_{\text{mint}} + g_{\text{transfer}} + 10^{-6}\ \text{USDC},$$
$$c_{L9} \approx g_{\text{mint}} + 3\,g_{\text{transfer}} + 3 \times 10^{-6}\ \text{USDC} + \$10\ \text{held},$$

where the gas terms are fractions of a cent and the $10 is parked, not spent. So the ladder certifies that money moved; it cannot certify that it mattered. The question is how much of the published rank rests on that floor.

First, the attack that would cost nothing at all. Anyone can make a stablecoin contract emit a Transfer event *from* any address by calling `transferFrom` with an amount of zero, which is how address-poisoning campaigns decorate their victims' histories. There are **0** zero-value rows in the counted set, so no rank rests on a spoof. Then the cheap ranks. Of 1,423 ranked identities, 60 (4.2%) have counted payments that sum to less than $1 and 127 to less than $10; the median ranked identity has paid $290. At the top the floor holds in practice: the cheapest of 22 L9 identities has paid $1,084.

Table 2. The addresses paid by the most ranked identities whose counted total is under $1.

Recipient	Kind	Identities	Minters	Minted	Payments	Total
`0xf907…5d3e`	EIP-7702 account	23	23	2026-03-01 to 2026-03-08	23	$3.45
`0x0e84…b808`	plain account	14	14	2026-02-16 to 2026-08-26	42	$0.12
`0x9305…f13f`	plain account	8	8	2026-05-21 to 2026-08-09	27	$0.27

The first row is the clearest picture of a rank bought for cents: 23 identities, each minted by its own wallet within one week in March, each making a single payment of $0.15 to the same address, $3.45 between them, and all 23 rank L7. We cannot say why. It might be a test, a farm, or a small operator provisioning agents, and small totals are not evidence of anything on their own: x402 micropayments are legitimately a fraction of a cent. What the table shows is that our lowest public rank can be reached for less than the price of a stamp.

A floor would be easy to write, for example a counted total of at least $1 to counterparties outside the identity's own minting cluster. It would also change a published definition that others may already cite, so we propose it for `rank.l7plus.v2` and will publish its effect on every ranked identity before switching. Until then two new metrics make the floor visible: `ranked_under_1usd` and `ranked_l9_min_counted_usd` (`rank.costtofake.v1`).

5. From mint to first payment

Most registrations never pay anyone, and the ones that do are not a random sample. We measured the time from each identity's mint to its first counted payment and estimated the probability of having paid by day t with the Kaplan–Meier product-limit estimator, treating identities that have not paid as censored at the time of measurement:

$$\hat{S}(t) = \prod_{i:\, t_i \le t} \left(1 - \frac{d_i}{n_i}\right), \qquad \hat{F}(t) = 1 - \hat{S}(t),$$

where d_i identities make their first payment at time t_i out of n_i still at risk. We split identities by the size of their minter's batch: how many identities the minting address created on Base in total.

Table 3. Probability of a first counted payment within t days of the mint, by minter batch size (Kaplan–Meier).

Minter's batch	Identities	Paid	1 day	7 days	30 days	90 days	365 days	Median wait
Minted alone	19,462	1,329	3.70%	4.26%	6.09%	7.07%	7.27%	0.5 d
2 to 9	26,834	91	0.07%	0.13%	0.23%	0.42%	0.51%	11.5 d
10 to 99	7,436	24	0.12%	0.20%	0.23%	0.31%	0.33%	4.8 d
100 to 999	8,408	7	0.01%	0.02%	0.02%	0.05%	0.09%	60.9 d
1,000 or more	33,686	175	0.07%	0.18%	0.36%	0.62%	0.66%	16.9 d
All	**95,826**	**1,626**	0.81%	0.98%	1.47%	1.82%	**1.91%**	1.8 d



Figure 1. Probability of a first counted payment within 365 days of the mint, by minter batch size.

Two things stand out. First, the gap: an identity minted on its own has a 7.3% chance of paying within a year, against 0.66% for one minted in a batch of a thousand or more, a ratio of about 11 to one. Batch minting is how platforms provision identities, and most provisioned identities are never used to pay. Second, the timing: those that pay mostly pay at once. The median wait is 1.8 days, and 47% of all identities that have paid did so within a day of the mint. An identity that has been silent for a quarter is very unlikely to start. For anyone counting "agents", this is the quantitative version of Note 01's warning that a registration is a name.

The estimate has two limits. The first payment is the first *counted* payment, so an identity that pays from a wallet we have not bound is invisible to it. And the payer's scan still has a backlog (Section 6), which means some first payments that have already happened are not yet in the table; the curves can only rise as it clears.

6. Two populations

ERC-8004 and x402 are often described as two halves of one agent economy: a registry of who agents are and a rail for how they pay. If that were true today, the wallets paying for machine services over x402 would be, to a large extent, the wallets bound to registered identities. We measured it from the sellers' side, where the chain gives a complete answer for the sellers listed in Coinbase's x402 Bazaar: every stablecoin transfer into their receiving addresses.

Table 4. Stablecoin transfers into x402 Bazaar sellers' receiving addresses, by network, over the window each network has been read. "Bound" means the payer is a wallet bound to an ERC-8004 identity on Base.

Network	Window (UTC)	Transfers	USD	Payers	Bound payers	USD from bound
Base	2026-09-20 to 2026-09-27	233,899	$233,004	4,666	288	$11,226
Arbitrum One	2026-09-23 to 2026-09-27	836	$117,368	806	0	$0
Polygon PoS	2026-09-24 to 2026-09-27	701	$77,626	667	0	$0
Solana	2026-09-20 to 2026-09-27	1,894	$336	77	0	$0

On Base, 288 of 4,666 payers (6.2%) are ERC-8004 wallets and they paid $11,226 of $233,004 (4.8%). The live metric uses a rolling seven days rather than the table's window, and read 5.0% of dollars at the snapshot. On the other networks the column is empty by construction, because we bind identities only on Base. The two populations barely meet: most machine buyers never registered an identity, and most registered identities never buy over x402.

We first tried to answer the same question from our own payment ledger, joining counted payments to the sellers' addresses, and got a figure so small it would have been a headline. It was wrong, and the reason is worth stating. Of the 247 bound wallets that paid a Bazaar seller in the seven days to the snapshot, 188 had never been read by our per-wallet payment scan: the three-hourly sweep had flagged them, and the scan, which reads whole histories and is rationed by our data budget, had not reached them. Across all bound wallets, 7,505 flagged wallets were waiting on 2026-09-27. Until that queue clears, ledger-side joins undercount, and every figure built on the counted set is a floor. We publish the seller-side shares as live metrics (`x402.bound.payers.v1`) and not the ledger-side one.

7. The recipients we could not name

Before this work, 88.0% of counted value went to 1,026 recipients of which the labelled ones carry evidence: a verified contract name, or receipts showing where the money went next. We took the 40 largest unlabelled recipients and read each from the chain: 26 are contracts, 10 plain accounts and 4 accounts delegated under EIP-7702. We never guess a plain account. Contracts get a label only if the explorer holds verified source for them, or for the implementation behind a standard EIP-1967 or beacon proxy slot, and the label says what the code is, never who runs it.

Table 5. Recipients labelled on 27 September 2026 from verified source, largest first.

Label	Category	Basis	Counted USD
UsdcVaultL2 (USDC vault) `0x3128…e858`	yield vault	verified implementation	$114,527
PropAMMAdapter (swap) `0x6a83…1102`	swap	verified name	$99,242
Aave v3 aToken (ATokenInstance) `0x4e65…c0ab`	lending	verified implementation	$30,351
0x Settler (BaseSettler) `0x7747…2359`	swap	verified name	$21,651
PancakeV3Pool (swap pool) `0x72ab…2d38`	swap	verified name	$18,403
OpenRouter (swap router) `0x50cf…5c4a`	swap	verified name	$16,318
Relay: RelayRouterV3 `0xb92f…ff4f`	bridge	verified name	$16,014
deBridge DLN (DlnSource) `0xef4f…eb66`	bridge	verified implementation	$13,981
Compound v3 Comet (CometWithExtendedAssetList) `0xb125…eb2f`	lending	verified implementation	$12,425
DexAggregatorCore (swap) `0x09ad…4afa`	swap	verified name	$11,458
Fluid Liquidity (FluidLiquidityProxy) `0x52aa…e497`	lending	verified name	$10,617
RiseVault (vault) `0x0109…84fd`	yield vault	verified implementation	$10,158
CCTPv2WithExecutor (bridge) `0x5289…f7d9`	bridge	verified name	$8,998
Wasabi vault (WasabiVaultRecoveryV1) `0x1c4a…8b24`	yield vault	verified implementation	$7,599
Moonwell market (MErc20Delegator) `0xedc8…6c22`	lending	verified name	$6,672
CLPool (concentrated-liquidity swap pool) `0x160d…eb12`	swap	verified name	$6,149

One of them is a small lesson in why names are not evidence. A contract verified under the name *OpenRouter* received $16,318 from 17 ranked agents. Read by name, it would have been the first sign of agents paying for AI model inference on-chain. Read by source, it is a swap router built on an allowance holder, Permit2 transfers and an `exec` entry point, and has nothing to do with the model API of the same name. We labelled it a swap.

Three held back: a perpetual-futures *TradingStorage* contract ($9,722 from 10 agents), because collateral for leveraged trades fits none of our categories yet; a contract verified as *EndaomentPSM* ($19,999), whose source we could not read well enough to say what the conversion does; and unverified contracts, the largest of which received $26,915 from 21 agents. With the new labels, 92.3% of counted value is labelled.

8. A theft replayed against our own detector

Our pages cite one real theft from an agent wallet: on 4 May 2026 a prompt hidden in Morse code led Grok's Bankr wallet on Base to send a large amount of the DRB token to an attacker, about $175,000 at the time according to Ledger. Our drain detector is the answer those pages give to such thefts. A rule is only worth publishing if we know what it misses, so we replayed the theft against it, transaction by transaction.

8.1 The rule, as the code implements it

The detector flags a counted payment t from a wallet whose earlier counted payments are $s \prec t$ when

$$v(t) \geq 5 \max_{s \prec t} v(s) \;\wedge\; v(t) \geq \$250 \;\wedge\; \mathrm{to}(t) \notin \{\mathrm{to}(s) : s \prec t\} \;\wedge\; |\{s \prec t\}| \geq 3,$$

looking back thirty days. The constants were calibrated on 24 September against every counted payment we then held: ten times and $500 fired on nobody, three times and $100 on thirteen agents, five times and $250 on five, few enough for a person to read every hit.

8.2 The chain

Table 6. The theft and the first twenty minutes after it, from the chain.

Time (UTC)	What moved
2025-03-07	The wallet `0xb105…e4f9` sends 1,179,283,962 DRB to `0xb6bc…ea68`: its only earlier DRB transfer.
2026-05-04 06:49:01	The theft, `0x6fc7eb7d…739a`: 3,000,000,000 DRB (3% of the 100 billion supply) to `0xe8e4…686b`, sent by the wallet itself.
2026-05-04 06:51:03	The attacker's address forwards all 3 billion to `0x35dd…6d19`, an account delegated under EIP-7702.
2026-05-04 06:57:57	50,337,005 + 12,537,001 + 567,126,058 DRB leave in three transfers.
2026-05-04 07:06:39	2,370,000,243 DRB (79% of the stolen amount) move in one transfer to `0xdd8e…d6de`.

Nothing came back to the wallet itself: the 433 million DRB it received afterwards came from an LP locker contract, which is fee income. Ledger reports that the Bankr team recovered the funds; other reports put the return at about 80%. The single transfer of 79% seventeen minutes after the theft is consistent with the second account, though the chain alone cannot tell us who controls the receiving address.

8.3 Three misses

We were not watching the wallet. Neither the wallet nor the attacker's address is bound to an ERC-8004 identity, and the ledger follows only bound wallets. **It was a token, not a stablecoin.** The ledger reads USDC, USDT, EURC and DAI from their known contracts, so a DRB transfer never becomes a row. **The size test would not have fired.** Even a token-aware version of the rule,

comparing like with like, sees 3,000,000,000 against a previous largest of 1,179,283,962: 2.54 times, under five. Strictly, the history test fails as well: the rule wants at least three earlier payments as a baseline, and the wallet had moved DRB once. The other two conditions held, and easily: the amount was far above $250, and the wallet had never paid that address.

A token-aware rule has a trap of its own, visible in the same history. Of the 30 transfers our index attributes to the wallet before the theft, 11 are of a token whose symbol reads "WETH" with a Cyrillic T: a spoofed token whose contract can emit transfer events from any address it likes. A detector that reads every token would page on fabricated events unless it restricts itself to contracts it trusts, which is exactly why the stablecoin ledger reads four contracts and nothing else.

And speed would not have saved the wallet. The stolen tokens were forwarded in two minutes and split within seventeen; our sweep runs every three hours and the detector hourly. A ledger explains a drain after it happens. Stopping one is the job of the wallet's own policy, set before the agent reads anything: spending limits, allow-lists, a second signature above a threshold. We have written that next to the rule on our drains page, and a token-aware variant for watched wallets, with a contract allow-list, is on our list for Note 03. It is not live.

9. Three corrections to our own record

The Bankrbot recovery. Six of our pages, the home page among them, and our llms.txt said the funds "were later recovered". The sources disagree about how much, and the chain shows 79% moving to one address; the pages now say that most of it was returned and give the on-chain account.

The drain rule in our API description. The machine-readable description of the `drain` pack said the rule compared a payment with the wallet's *median*. The detector compares it with the *previous largest* counted payment, as every page and the code say. The description has been corrected.

Wallet kinds. Our wallet table recorded every one of 48,824 wallets as an ordinary account; the column was set on insert and never read from the chain. In a random sample of 150, 24 carried an EIP-7702 delegation and 6 were contracts. No published figure read that column, but our own first analysis of the Adapter's depositors did (Section 2.3). A worker now reads each wallet's code, 1,200 wallets per run; 500 had been read when this note was written.

10. What changed on the site

Table 7. New live metrics, published every three hours, with their definitions.

Metric	Definition	Value at the snapshot
observed_pay_usd_own_positions	pay.usd.ownpositions.v1	$6,017,368
observed_pay_usd_left_control	pay.usd.leftcontrol.v1	$3,480,064
pay_reciprocal_pairs	pay.loops.v1	0
pay_agent_to_agent_usd	pay.loops.v1	$28,404
ranked_under_1usd	rank.costtofake.v1	60
ranked_l9_min_counted_usd	rank.costtofake.v1	$1,084
survival_paid_365d_all	rank.survival.km.v1	1.91%
survival_paid_365d_single	rank.survival.km.v1	7.27%
survival_paid_365d_burst1000	rank.survival.km.v1	0.66%
survival_median_days_first_pay	rank.survival.km.v1	1.8 d
x402_base_payers_7d	x402.bound.payers.v1	4,426
x402_base_payers_bound_7d	x402.bound.payers.v1	247
x402_base_payers_bound_share_7d	x402.bound.payers.v1	5.6%
x402_base_usd_bound_share_7d	x402.bound.payers.v1	5.0%

The money page, the home page and the press kit now bind the own-positions split live beside the counted total. The drains page carries the replay. Sixteen new counterparty labels are on `/api/partners` with their evidence, and the findings feed carries five new entries: own positions, the managed fleet, the price of a rank, survival by batch, and the two populations.

11. Reproducibility

Every figure here is reproducible from the chain and from the endpoints the ledger publishes. The own-position test needs only the receipt of each transaction; the loop measures are a self-join of counted payments; the survival table needs each identity's mint block from the ERC-8004 Identity registry (`0x8004…a432`) and the first counted payment; the x402 shares need the Bazaar's seller list and the stablecoin transfers into those addresses. In SQL against our schema, the split in Section 2 is

```
SELECT sum(amount_usd) FILTER (WHERE own_position) AS own_positions,
       sum(amount_usd) FILTER (WHERE own_position IS NOT TRUE) AS left_control
FROM money_events
WHERE kind = 'PAY' AND NOT pre_registration AND NOT pass_through
  AND pass_checked_at IS NOT NULL;
```

and, in simplified form, the reciprocity in Section 3 is

```
WITH e AS (SELECT from_wallet a, to_address b, sum(amount_usd) usd
           FROM counted_payments GROUP BY 1, 2)
SELECT count(*), sum(least(x.usd, y.usd))
FROM e x JOIN e y ON y.a = x.b AND y.b = x.a WHERE x.a < x.b;
```

Live values: `agenticfinancegraph.com/api/state`; definitions: `/def/<id>`; labels with evidence: `/api/partners`; findings: `/api/findings`.

12. What this note does not assert

- That the managed fleet in Section 2 is run by any particular company, or by one operator at all.
- That an identity with a small counted total is fake, or that a large one is a business.
- That identities which have not paid are idle: they may pay from wallets we have not bound, or on chains we do not read.
- That x402 buyers without an identity are not agents.
- That our detector, or any ledger, can prevent a theft. It can describe one afterwards, with evidence, and say what it missed.

References

1. Agentic Finance Graph (2026). *The Binding Record: how we join an AI agent's identity to the money it moves.* Research Note 01, v2.1, 27 September 2026.
2. ERC-8004: Trustless Agents. Ethereum Improvement Proposals.
3. x402: an HTTP-native payment protocol, and the x402 Bazaar discovery catalogue. Coinbase Developer Platform documentation.
4. Safe smart account v1.4.1; ERC-7579 modular smart accounts and the Safe7579 adapter; ERC-4337 account abstraction (EntryPoint v0.7).
5. Aave v3 protocol documentation; EIP-4626 tokenized vaults.
6. Kaplan, E. L. and Meier, P. (1958). Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association* 53(282), 457–481.
7. Reports of the 4 May 2026 Grok/Bankrbot prompt-injection theft: Ledger; OECD.AI incident record 2026-05-04; Giskard, "How Grok got prompt-injected"; The Crypto Times, 4 May 2026.

Written by N. Simonyan for Agentic Finance Graph, with AI assistance in analysis and drafting. All measurements are the ledger's own and are reproducible from the chain and the endpoints above at the stated time. Figures in this note are frozen at 2026-09-27 14:52 UTC; the live metrics move as the payment scan and the receipt check clear their queues.