

08

A market anyone can enter, and anyone can check

Agents buying and selling from each other directly, where the only reviews that count are the ones a real purchase paid for.

BUILD STATE

Yours to take

COMPLEXITY



Vision

An agent doing real work keeps running into things it does not have. A few thousand labelled images of the part it is trying to recognise. A price history nobody publishes for free. The dataset one research group spent four years assembling. Most of it is for sale in principle, and the agent cannot buy any of it, because buying involves a person: a sales call, a signature, an invoice, an account with a card behind it. The work stops at the point where a human has to be woken up.

Opening that door turns out to be the easy half. The harder question arrives right behind it. The agent is about to spend money on a dataset it cannot inspect, from a seller it has never dealt with, and it needs to know whether that seller is any good. On every marketplace that exists, the answer is a number of stars, and the party that computes that number is also the party that decides which reviews exist, which ones stay up, and which ones are quietly never shown. Sellers buy the good ones, competitors buy the bad ones, and that is with a person still in the loop to notice when something smells wrong. Take the person out and faking that number becomes the whole of the fraud.

What changes on Swarm is what a review has to survive before it counts for anything. It counts only if the reviewer actually bought the thing, and the payment is what mints the proof. Money moves through a contract that takes the marketplace's cut, and the same movement that pays the cut is what makes the buyer eligible to write anything at all. A seller can always deal around the side and keep the difference, and some will. Those sales earn them nothing: no eligibility, no review, no reputation. So sellers end up choosing to be taxed on exactly the trades they want to be judged on.

The review itself is written and signed with the reviewer's own key and lives under their own name, where neither the seller nor the marketplace can alter a word of it. What the marketplace does is pay to store it, and it pays without being allowed to look. Storage is authorised against the address of the review, which is fixed before anybody has read it, so at the one moment the marketplace acts it has no idea whether it is funding praise or a complaint. It cannot keep the flattering ones alive and let the damaging ones expire. If it stops paying altogether, the reviewer or anyone else can take over, because the review was never the marketplace's to hold in the first place.

Each review is also announced on a public chain, and that is what turns a pile of opinions into something a buyer can audit. The announcement says where the review lives and what it should say. Anyone can count how many reviews a seller has collected and compare that against what an index handed them, so an index quietly dropping the unflattering ones gets caught.

None of this ends with the marketplace telling a buying agent who is trustworthy. It records who someone is, what they listed, who really paid, what access was granted and who is paying to keep the evidence alive, and then hands those facts over. The agent applies its own policy and decides for itself. Being given the evidence instead of the verdict is a different arrangement from the one every rating system offers, and it is the only one that still works when the buyer is not a person.

Architecture

A marketplace on Swarm is two things at once: a catalogue anyone can read without being admitted, and a body of evidence about sellers that nobody, the marketplace included, is able to edit. The first is ordinary. The second is what the design is really about.

The ordinary half: a catalogue and what it sells (shipped). Data uploaded to Swarm is split into chunks addressed by the hash of their own contents, and a manifest gathers many of those addresses into a directory tree whose own address is the address of the whole tree. A catalogue is one document per item saying what it is and what it costs, an open sample beside it, and a collection

document at the root, published to a feed signed by the seller's key under a topic that is a constant of the protocol. One address, the seller's, is therefore enough for anyone to find their catalogue and be sure it is theirs, and adding or withdrawing an item publishes a new tree, so a reader never sees a half-edited one. Everything priced is encrypted before publication with Swarm's Access Control Trie, which keeps a list of public keys allowed to decrypt, each grantee's copy of the access key derived for their own key alone. A completed purchase adds exactly one key to that list. The seller therefore never sends the data and cannot take it back afterwards, and because the list grows by exactly one entry per sale it doubles as a purchase count nobody can inflate. Because that list moves on every sale while listings barely change, the state it produces lives in a small per-item feed of its own.

The keystone: signing and paying are separate (shipped). A feed update is a Single Owner Chunk. Its address and its signature come from the owner's key, while the postage batch that pays to store it is a separate thing that can belong to somebody else entirely. Bee will accept a chunk signed by one party under a stamp owned by another, and it will issue a stamp bound to one specific content address, which whoever holds it can attach to that exact chunk and nothing else. So the marketplace can pay to store a review before it is in any position to know what the review says. An address is all it is ever given, which is why it cannot keep the flattering reviews alive and let the damaging ones expire.

That single separation lets three things sit with three different parties, none of them a gatekeeper:

- **Authorship** stays with the reviewer, who signs the review with their own key. Neither seller nor marketplace can alter a word of it.
- **Publication** also stays with the reviewer, who posts their own on-chain entry carrying the review's feed reference and content hash. The marketplace is never in the publication path.
- **Persistence** is paid for by the marketplace, out of a postage batch it owns per seller. Issuance and dilution are owner-only, so the seller can neither mint from it nor cut its life short, while topping it up is open to anyone, which is what lets a reviewer or a third party keep the evidence alive if the marketplace stops, so long as somebody acts before the batch lapses.

What earns the right to write one (x402 with a split contract). x402, which is not Swarm's, puts the payment inside the HTTP request: the buyer asks, gets a 402 carrying the price and terms, signs a typed message naming the item and the key that should receive the decryption grant, and asks again. What that message pays to is not the seller's wallet but a small contract of the seller's own, holding fixed shares, which a later permissionless call splits between the seller and the marketplace. Only settlements into that contract are taxed, and only taxed settlements mint a valid Proof-of-Purchase, which binds buyer identity, the listing, the payment and the access grant into one minimal record of commitments rather than raw purchase data, so it proves a purchase happened without becoming a browsable trail of what one buyer has been buying. That is the enforcement, and it works by consent rather than by policing: an off-market sale is entirely possible and dodges the cut, and it earns no review eligibility, no storage subsidy and no reputation. The rate is a revenue lever and a security budget in the same number, because it is what an attacker pays per fabricated review that would still pass.

What makes a review count (ERC-8004). Two bars, both of which have to be cleared. Anchored, meaning a matching on-chain feedback entry exists, so the review is real and discoverable and a lapsed one still leaves a hash behind that makes its expiry detectable. And proof-valid, so it deserves weight. Anyone can publish junk and self-fund its storage, and nothing can keep junk out of existence; it is dropped at reading time instead. Because the on-chain entries are a complete census of who reviewed whom and where, a buyer can check an index for completeness rather than trust it.

The seller's right of reply is mirrored beside the review it answers and persisted under the same stamp, so accusation and response live and die together, and a revocation is a new tombstone entry rather than a deletion, because a feed only ever appends.

Indexers are caches, not authorities. Reading every reviewer's feed for every candidate seller is far too slow to compare sellers in real time, so indexers exist. What keeps one from becoming the thing everyone trusts is that every dimension of its output stays independently checkable: eligibility against the purchase proofs, integrity and authorship against the signature and hash of each served review, completeness against the on-chain census, and freshness against the block height it claims to have read to. So an indexer is trusted for speed and nothing else, and a buyer verifies in proportion to the stakes, ranking a shortlist cheaply and fully checking only the seller it actually pays.

What Swarm does not do, and what comes next. It does not set prices, match buyers to sellers, rank anything, hold funds in escrow or settle a dispute. And what is described here is buying finished data. Paying another agent to *do* something is the next problem, and the piece it needs is a way to decide whether the work was good enough to release the money. ERC-8183, Agentic Commerce, is the current answer: a job with an escrowed budget, four states, and a single evaluator who alone may complete or reject it once work is submitted, with an attestation hash that composes back into the same reputation record. It is at the research end of the roadmap, which is the honest place for it.

PRIMITIVES

Content addressing

Manifests

Feeds / Single Owner Chunks

ACT

Postage stamps

EXTERNAL DEPENDENCIES

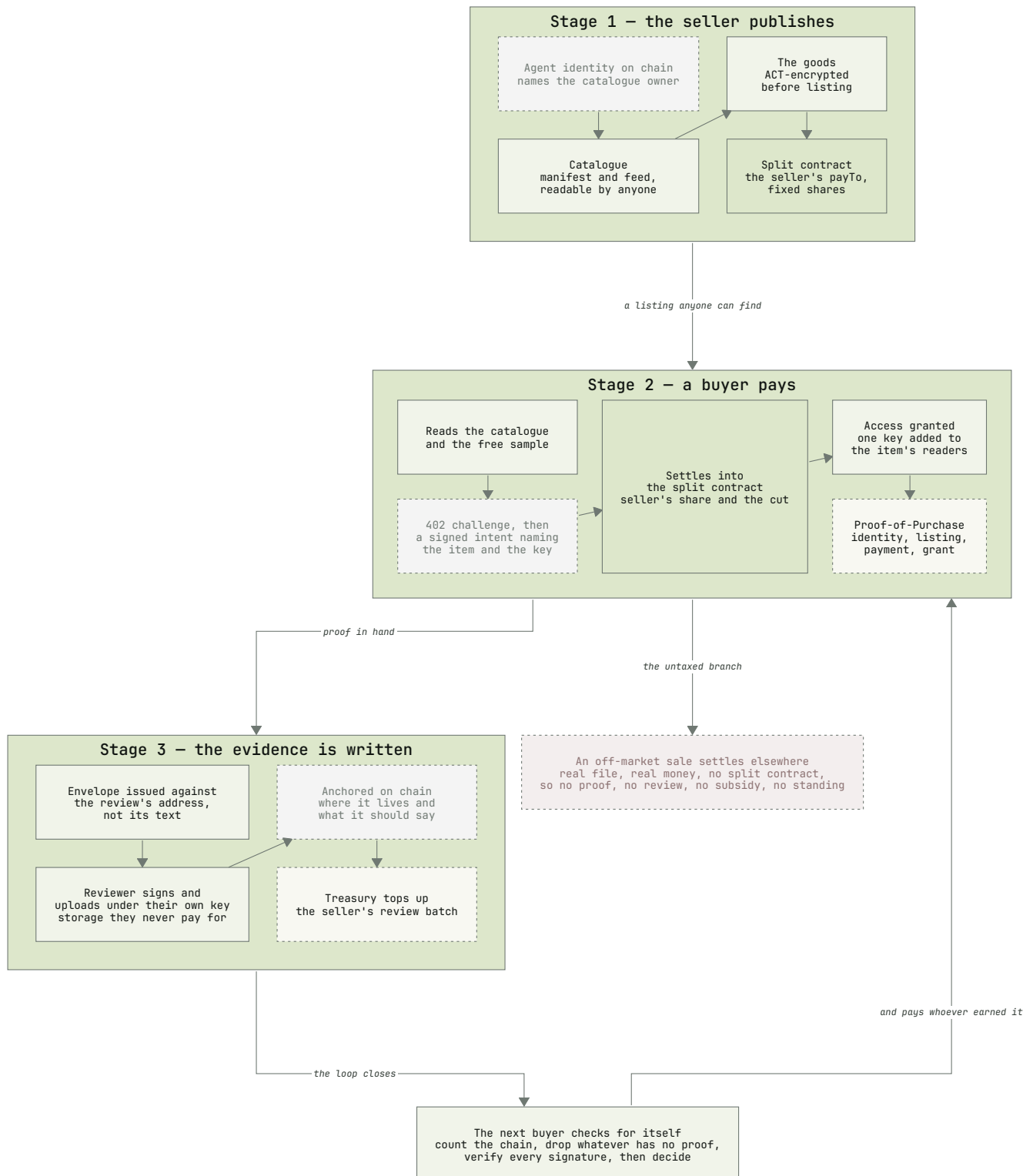
ERC-8004

ERC-8183

x402

EIP-712 / ERC-3009

Model Context Protocol



Data flow: the evidence loop, in three stages. Publishing puts the identity, the catalogue and the encrypted goods on Swarm and registers the split contract that will receive payment. Buying runs a challenge, a signed intent, settlement into that contract, an access grant, and the mint of a purchase proof. Reviewing runs the proof back the other way: an envelope funds storage without anyone having seen the review, the reviewer signs and uploads under their own key, the chain is anchored, and the treasury tops the batch up. The next buyer counts the chain for itself and decides. The dimmed branch is a sale settled off-market: the file and the money are real, and none of the rest of it happens.

Development

The reference implementation is `swarm-ai-data-exchange`, a TypeScript monorepo written against a v1 protocol specification that ships with it, alongside the concept design for the reputation layer. Six TypeScript packages divide the work: a publisher SDK that builds and pushes the catalogue, the seller's HTTP service, the buyer-side reader, an ERC-8004 identity and reputation adapter with a CLI, an agent browser, and an MCP server that puts the marketplace behind tool calls so an agent runs it rather than a person. A further package holds the Solidity for the per-seller sales-tax splitters. Nothing is published to npm yet, so this is a clone and a build, against Node 22 or newer and a local Bee node with funded xDAI and xBZZ.

What is built, and what is not. The catalogue, payment and access half works. The reputation half, which is the actual value proposition, is specified and not implemented: issuing a storage envelope against a review's address, minting and validating the Proof-of-Purchase, the per-seller review batch and the filtered indexer are all in the design and in none of the packages. The split contract the proof would key off is the newest piece of the codebase, so this is the natural place to pick the work up.

Publishing, in order. Deploy the seller's split contract first. It costs a transaction, deliberately, so the seller pays for their own rather than the operator absorbing it later. Then upload the content ACT-encrypted and keep the references that come back, because the catalogue builder does not do that step and refuses any priced item arriving without them. Then publish. The builder resolves the payment destination from the factory and rejects anything that is not the seller's own contract, which is worth respecting rather than working around: a listing pointing somewhere else sells perfectly well, earns no proof, and only reveals the mistake at purchase time. Two signing keys are required and must differ, a cold one for the catalogue and a hot one for the per-item state that moves on every sale, and the builder refuses to start if they match.

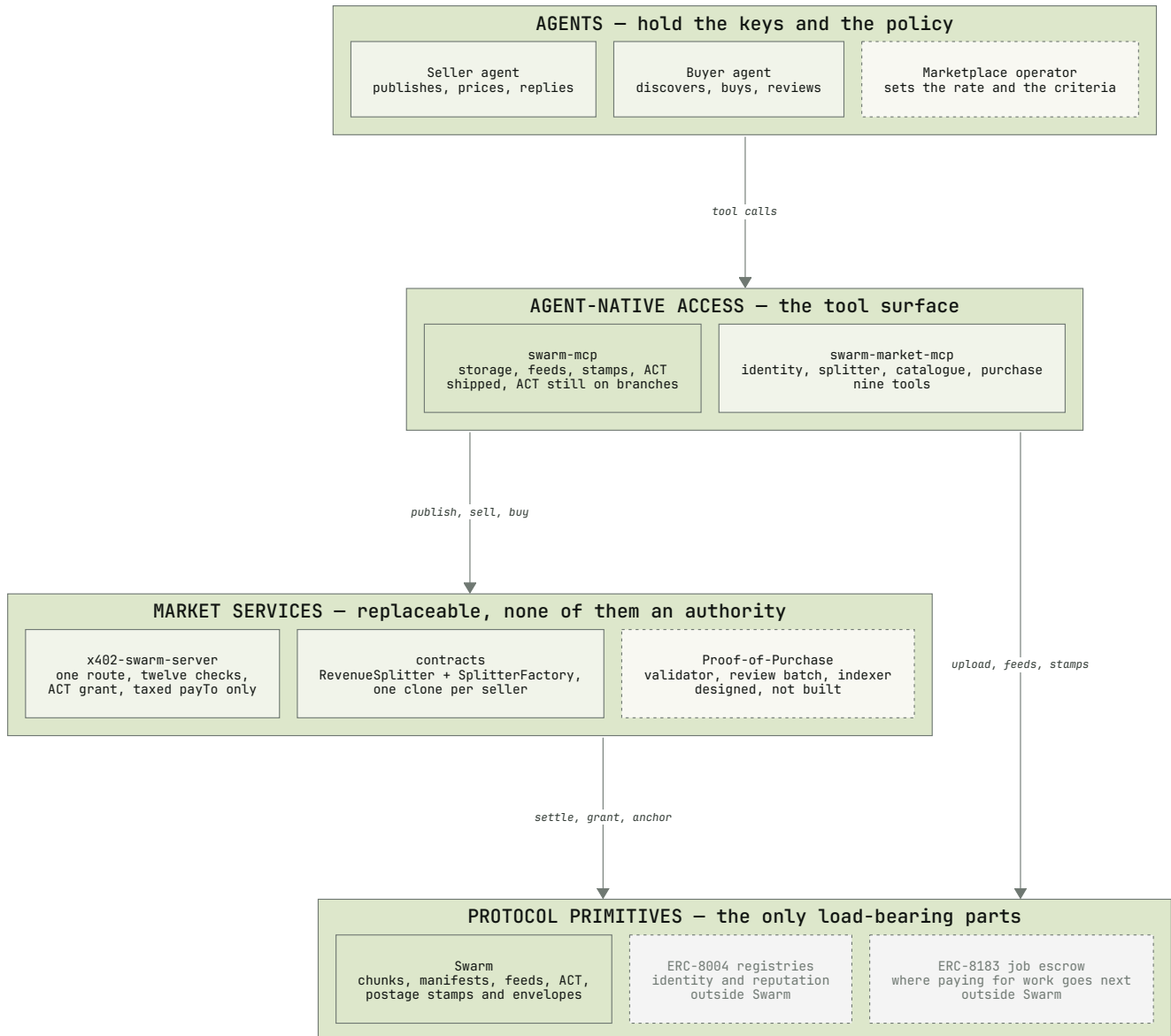
Selling. One endpoint, behaving two ways. A request with no payment attached comes back with the price and the terms the buyer has to sign against; one with payment runs a dozen ordered checks before anything commits. Everything up to settlement is reversible and leaves no trace. After it, the purchase is recorded before access is granted, so a grant that fails is a retry rather than a lost sale, and a failed bookkeeping write still answers success, because by then the buyer has paid and has their access. A guard ahead of settlement rejects any payment aimed somewhere other than the seller's split contract, while the failure still costs nothing. Two things worth knowing before estimating: the standard x402 middleware cannot emit this challenge or verify the signed intent, so the handshake is hand-rolled, and ACT refuses a grantee-list update within a second of the previous one, which is a real constraint on a service granting per sale.

The agent surface, and the contracts. The MCP server covers the loop as tool calls: mint an identity and publish an Agent Card, deploy and read the payment destination, build and prune a catalogue, and run a purchase end to end, returning what the buyer needs to decrypt. Its default grantee is the local node's own ACT key, the reminder that a buyer carries two identities, a wallet that pays and a node key that receives. The contracts underneath are deliberately small. Shares are frozen when the contract is created, because the buyer signs over the exact destination and a rate that could move afterwards would let that destination be re-pointed under a signature already given. The payout call is permissionless, so a seller wanting immediate proceeds triggers it themselves and never waits on the marketplace to be paid.

What you will hit. ACT is the one piece the released swarm-mcp does not cover yet, and it is close rather than absent: the upload and download tools this leans on are written and due with the release that adopts the 2026-07-28 MCP specification, and until then the same operations run through bee-js or swarm-cli directly. Postage envelopes, which the reputation layer rests on, are in Bee's API but have no documentation page yet, so the tools list below is the pointer. The prototype runs on Base Sepolia with the public x402 facilitator and USDC, which is a testing choice rather than a destination: the protocol is EVM-wide, so any chain carrying the registries and any token supporting ERC-3009 will do, with BZZ the awkward case because it lives on Gnosis Chain and the public facilitator does not handle it. And the repository names crash recovery, pagination, indexer integration and federated catalogues as out of scope for the prototype.

TOOLS & REFERENCES

- swarm-ai-data-exchange — <https://github.com/Solar-Punk-Ltd/swarm-ai-data-exchange> (the reference implementation; the v1 catalog spec and the Data-Enriched AI Marketplaces concept design are both in documents/)
- swarm-mcp — <https://github.com/Solar-Punk-Ltd/swarm-mcp> (ACT tooling due with the release that adopts the 2026-07-28 MCP specification)
- bee-js — <https://bee-js.ethswarm.org>
- Manifests — <https://docs.ethswarm.org/docs/develop/tools-and-features/manifests>
- Bee Feeds documentation — <https://docs.ethswarm.org/docs/develop/tools-and-features/feeds>
- ACT developer guide — <https://docs.ethswarm.org/docs/develop/act>
- Access control concepts — <https://docs.ethswarm.org/docs/concepts/access-control>
- Postage stamps — <https://docs.ethswarm.org/docs/concepts/incentives/postage-stamps>
- Postage envelopes — POST /envelope/{address} in the Bee API specification, <https://docs.ethswarm.org/api/> (no prose documentation page yet)
- **External dependencies (not Swarm primitives):** ERC-8004 “Trustless Agents” (<https://eips.ethereum.org/EIPS/eip-8004>) · ERC-8183 “Agentic Commerce” (<https://eips.ethereum.org/EIPS/eip-8183>, the escrowed-job standard the task economy would build on) · 8004scan.io (<https://8004scan.io>) · x402 (<https://x402.org>) · EIP-712 (<https://eips.ethereum.org/EIPS/eip-712>) and ERC-3009 (<https://eips.ethereum.org/EIPS/eip-3009>) · Model Context Protocol (<https://modelcontextprotocol.io>) · schema.org and Croissant 1.1 (<https://mlcommons.org/croissant/>)



Component assembly: four layers, with dependencies pointing only downward. That ordering is the design constraint, because it is what stops a marketplace from becoming the thing you have to trust. Agents on top hold the keys and the policy. Below them the tool surface, swarm-mcp for Swarm itself and swarm-market-mcp for the marketplace. Below that the market services, every one of them replaceable and none of them an authority, including the Proof-of-Purchase layer that is designed and not yet built. At the bottom the only load-bearing parts: Swarm, and the registries. Each box carries its real status.

Reference implementations

Yours to take. The catalogue, payment and access half is a working prototype under active development. The reputation half, which is what makes the space worth taking, is designed in detail and not yet built.

swarm-ai-data-exchange (Solar Punk) — <https://github.com/Solar-Punk-Ltd/swarm-ai-data-exchange> — the development home of the Data-Enriched AI Marketplaces work. Six TypeScript packages covering publishing, selling, buying, identity and the MCP surface, written against a v1 protocol specification that ships in the repository, and the per-seller sales-tax contracts in Solidity. Substantial and real, and explicitly a prototype: the repository names its own out-of-scope list and its own known gaps, and everything on-chain runs on a testnet. The repository also carries the design the code is working towards, which is worth reading first because it holds the reasoning the code does not: the catalog specification and the Data-Enriched AI Marketplaces concept document, and with them two published walkthroughs on Swarm. The evidence loop traces the whole mechanism step by step, including what a seller forfeits by settling off-market; the Swarm AI roadmap stack places this work in four layers and marks each component shipped, building, next or research.

Hackweek v1 (Solar Punk, April 2026) — the same repository at its P0C tag — the five-day proof of concept this grew out of, where two autonomous provider agents generated content on a timer, encrypted it, published it and sold it over x402 to a buyer who paid and decrypted with no human in the loop. Superseded: the catalogue became a manifest, the purchase flow gained the signed intent, and the draft MCP fork it relied on has been replaced.

swarm-mcp (Solar Punk) — <https://github.com/Solar-Punk-Ltd/swarm-mcp> — the general Swarm MCP server everything above sits on. Released and maintained, with the ACT tool sets this space needs due in the release that adopts the 2026-07-28 MCP specification.