DESIGNED BY DMITRY SERGEEV

AGENT COMMERCE

SOFTWARE ARCHITECTURE

13.09.2026

Table of contents

Executive summary 2

Functional requirements 5

Architecture decisions 7

Technical architecture - C4 12

Quality attributes and SLOs 22

Threat model - STRIDE 25

Deployment, Delivery, Recovery and Compliance 28

Delivery sequence and architecture gates 32

Executive summary

Agent Commerce Gateway sits in the merchant's infrastructure in front of an existing HTTP API. A merchant describes a capability, its input, its price, its backend endpoint, and the protocols through which it may be exposed. Buyer agents discover and invoke that capability without requiring the merchant to embed agent protocols or payment code in the backend.

The gateway is a **deterministic commerce control plane**, not an AI model, wallet, custodian, marketplace, or merchant of record. Protocol-specific input is translated into a canonical request and every request traverses one execution pipeline. That pipeline validates input, resolves price, enforces payment, reserves replay protection, settles where required, invokes the merchant backend, and records a receipt. A protocol adapter is never allowed to call the backend or settle a payment directly.

The initial release exposes native HTTP and MCP, enforces x402 v2 payments, settles EIP-3009 USDC transfers on a local EVM chain, Base Sepolia, or Base, and stores receipts, events, and payment attempts in SQLite. The gateway sends funds directly from buyer to the merchant-controlled destination. Production deployments use a remote facilitator; the gateway holds neither buyer nor merchant keys.

Extended scope preserves the same core boundaries:

- A2A, ACP, and UCP enter through protocol adapters.
- MPP enters through the payment-provider interface.
- AP2 adds mandate verification before settlement and provenance to the receipt; it is not a separate transport or payment rail.
- OpenAPI import generates canonical resource configuration rather than a second execution path.
- PostgreSQL replaces the storage adapter where high availability or multi-replica operation is required.
- Richer observability and a `doctor` GitHub Action operationalize the same descriptors and health checks exposed by the running gateway.

> **Core security invariant.** A paid resource may be delivered only after its payment proof has been verified, its authorization identity has been durably reserved against replay, and settlement has succeeded. Every protocol must reach that decision through the same pipeline.

Goals

- A2A, ACP, and UCP enter through protocol adapters.

- Keep the merchant in control of infrastructure, pricing, backend behavior, receipts, and settlement destination.
- Make protocol and payment-rail support additive at narrow interfaces.
- Make the no-payment, rejected-payment, uncertain-settlement, paid-but-not- delivered, and delivered outcomes distinguishable and auditable.
- Provide deterministic local development and conformance tests without a hosted model, public network, real funds, or hosted facilitator.
- State protocol revisions and unsupported capabilities explicitly.

Non-goals

- Custody of buyer funds, merchant funds, production private keys, or seed phrases.
- Acting as a payment service provider, merchant of record, marketplace, or dispute arbiter.
- Buyer identity, KYC, sanctions screening, fraud scoring, tax calculation, refunds, chargebacks, fulfillment, or post-delivery order management.
- General multi-tenancy or shared merchant data in one gateway deployment.
- Dynamic pricing in the initial release.
- Long-running agent tasks, subscriptions, streaming micropayments, batched settlement, or asynchronous delivery in the initial execution model.
- A production edge, certificate authority, secret manager, RPC provider, monitoring service, or backup service operated by this project.

Delivery scope

PHASE	SCOPE	EXIT CONDITION
Authorization expansion	Configuration and CLI; native HTTP; MCP tools; canonical pipeline; free and fixed-price resources; x402 v2 exact on EVM/EIP-3009; local Anvil, Base Sepolia and Base; SQLite receipts/events/payment attempts; diagnostics; deterministic demo and dashboard	Paid and free calls pass conformance and end-to-end tests; a paid test proves exact buyer and merchant balance changes and a durable receipt
Protocol expansion 1	A2A JSON-RPC discovery and synchronous message/send	Official client discovers a skill and invokes free and paid resources through the same pipeline
Authorization expansion	AP2 SD-JWT VC + Key Binding mandate verification with offline did:key trust	Invalid, expired, undisclosed, untrusted, replayed, or cart-mismatched mandates fail before settlement; valid mandate reference reaches the receipt

Protocol expansion 2	ACP Checkout API, capability negotiation, adapter-owned session state, delegated-token pass-through	Full create/update/read/complete/cancel lifecycle; one completion produces one execution and one receipt; token is never persisted
Payment expansion	MPP Core plus charge intent and EIP-3009 on Base	Real MPP client completes challenge, settlement, delivery, and receipt through the payment-provider interface
Platform expansion	OpenAPI import; broader conformance; doctor GitHub Action; UCP design and implementation; Shopify/WooCommerce examples; PostgreSQL; metrics/traces/alerts	Each item has a versioned contract, tests, operational evidence, and an explicit support claim

Phase labels describe delivery order, not partial implementation: every row is a requirement of this architecture. UCP stays at a design gate until a revision, supported slice, discovery surface, state model, and conformance client have been selected.

Functional requirements

This section describes user and business outcomes. Protocol and storage mapping appears later and is deliberately excluded here.

ID	ROLE AND END-TO-END USE CASE	EXIT CONDITION
FR-01	Merchant - receive funds directly. Set a merchant-controlled destination and receive payment without funds passing through an account controlled by the gateway.	Adds agent sales without rewriting the merchant's business service.
FR-02	Buyer agent - discover capabilities. Ask the merchant what can be purchased and receive names, descriptions, expected input, price, and supported interaction choices.	Lets an agent decide whether and how to buy before invoking anything.
FR-03	Buyer agent - invoke a free capability. Submit valid input, receive the merchant result, and receive a correlation reference for the completed interaction.	Supports lead generation and free services through the same gateway as paid services.
FR-04	Buyer agent - buy a paid capability. Request a capability, receive an exact payment request, authorize payment to the merchant, retry, and receive the result only after payment succeeds.	Enables machine-to-machine pay-per-use commerce.
FR-05	Merchant - receive funds directly. Set a merchant-controlled destination and receive payment without funds passing through an account controlled by the gateway.	Avoids intermediary custody and preserves merchant control of revenue.
FR-06	Buyer and merchant - prevent duplicate use. A payment authorization that has already been used must not purchase a second delivery, even when requests race or the gateway restarts.	Prevents double delivery and payment disputes caused by replay.

FR-07	Merchant operator - investigate a transaction. Find the request, payment status, settlement reference, backend outcome, timing, and receipt by one correlation identifier.	Supports reconciliation, support, and incident analysis.
FR-08	Merchant operator - distinguish paid from delivered. When payment succeeds but the merchant service fails, see that payment is final and delivery failed rather than seeing a false rollback or generic failure.	Enables honest customer remediation and financial reconciliation.
FR-09	Merchant operator - verify readiness. Check configuration, storage, exposed interaction methods, payment routes, destination, and dependency health before accepting traffic.	Reduces revenue loss and misdirected-payment risk from bad deployment configuration.
FR-10	Merchant operator - protect commercial history. Allow only authorized operators to read receipts and events; if operator access is not configured, expose no ledger data.	Keeps revenue and customer-address history private by default.
FR-11	Application developer - embed the gateway. Compose the gateway, storage, interaction adapters, and payment providers inside another service while installing only the capabilities that service needs.	Supports lightweight adoption and controlled dependency footprint.
FR-12	Protocol client - verify support honestly. Read the exact supported revision, capabilities, health, and known unsupported features for each interaction or payment method.	Prevents clients from relying on marketing claims or silent partial support.
FR-13	Buyer agent - interact through another open standard. Discover and synchronously invoke the same merchant capability through a newly supported standard without changing merchant behavior or weakening payment enforcement.	Expands distribution without duplicating commerce logic.
FR-14	Human approver and buyer agent - prove delegated intent. Present a verifiable authorization that binds the buyer, merchant, selected capability, exact amount, audience, and interaction context; reject missing or mismatched authorization.	Adds human authorization evidence to autonomous purchasing.
FR-15	Buyer agent - manage checkout before purchase. Create, change, inspect, complete, or cancel a checkout session and ensure completion can produce at most one delivery.	Supports session-oriented agent checkout while preventing duplicate completion.
FR-16	Merchant - reuse its payment processor. Receive a delegated checkout credential only for the duration of completion, process it through the merchant's existing provider, and prevent the gateway from storing or charging it.	Preserves the merchant-of-record relationship and existing payment operations.
FR-17	Buyer agent - use an alternative machine-payment method. Complete a one-time charge over a second payment standard while receiving the same delivery and receipt guarantees.	Reduces dependence on a single payment rail.
FR-18	Merchant integrator - import an API description. Convert selected operations from an existing API description into reviewable capability definitions before publication.	Lowers onboarding cost while keeping the merchant in control of exposure and pricing.
FR-19	Platform operator - scale durable state. Move commerce history and replay protection to an operational database without changing client or merchant-facing behavior.	Supports backup, high availability, and larger transaction volumes.

FR-20	Platform operator - operate from measurable signals. Observe request rate, latency, payment outcomes, delivery failures, dependency health, queue pressure, and storage health, and alert on agreed objectives.	Makes production reliability and revenue-impacting failures visible.

Non-goals

Publish: describe capability -> validate input and destination -> review public discovery representation -> start gateway -> verify readiness -> announce endpoint.

Free invocation: discover -> choose capability -> submit input -> validate -> invoke merchant backend -> record receipt -> return result.

Paid invocation: discover price -> request capability -> receive payment challenge -> sign buyer-to-merchant authorization -> retry -> verify -> reserve against replay -> settle -> invoke backend -> record receipt -> return result.

Reconcile paid-but-undelivered: locate correlation id -> confirm settlement reference -> inspect backend failure -> remediate with the buyer outside the gateway's automatic flow; never pretend settlement was rolled back.

Session checkout: create session -> negotiate supported capabilities -> update cart -> review state -> complete once or cancel -> retain a single terminal audit outcome.

Architecture decisions

AD-01 - Self-hosted, single-merchant modular monolith

Context. Merchants need to expose existing services without surrendering traffic, transaction history, or settlement control to a central intermediary.

Alternatives. Hosted multi-tenant marketplace; a library embedded separately into every backend; a self-hosted gateway process.

Decision. One gateway deployment serves one merchant configuration. The runtime is a modular Node.js/TypeScript monolith with explicit ports for protocol adapters, payment providers, backend execution, storage, logging, clock, and identifiers.

Consequences. Deployment and trust boundaries remain understandable and the gateway can also be embedded as a library. Fleet management and tenant-level RBAC are outside scope. SQLite deployments run one writer process.

AD-02 - Canonical commerce model before protocol adapters

Context. MCP, A2A, ACP, UCP, and native HTTP have different discovery, request, error, and state shapes. Allowing those types into business logic would couple payment enforcement and receipts to each protocol.

Alternatives. Independent end-to-end implementations per protocol; a common wire format imposed on clients; a wire-neutral canonical model.

Decision. The core defines resource, pricing, request, payment requirement, payment result, execution outcome, receipt, event, and error concepts. Each adapter maps its wire representation into and out of those types. Provider-native challenges remain opaque data at the core boundary.

Consequences. A new adapter does not change core payment logic. Some protocol-native capabilities are intentionally unsupported because they do not fit the canonical synchronous invocation model.

AD-03 - One execution pipeline is the delivery security boundary

Context. Payment enforcement fails if any route or adapter can call a merchant backend directly.

Alternatives. Payment middleware per route; adapter-owned workflows; one central pipeline.

Decision. Every delivery traverses this ordered flow: resolve resource and exposure -> strip reserved payment input -> validate input -> resolve price -> create challenge or verify proof -> reserve replay key -> settle -> invoke bounded backend -> persist audit data -> return outcome. Adapters do not receive a backend executor or payment credentials.

Consequences. Payment enforcement becomes a system invariant. The pipeline is security-sensitive and changes to its ordering require adversarial tests and architecture review.

AD-04 - Non-custodial buyer-to-merchant settlement

Context. A gateway-held merchant wallet or buyer balance would add custody, key compromise, accounting, and regulatory risk.

Alternatives. Gateway wallet and later payout; hosted PSP account owned by the project; authorization that transfers directly to the merchant.

Decision. The payment requirement names a merchant-controlled destination. For x402 and the MPP charge path, the buyer signs an EIP-3009 authorization bound to recipient, value, validity window, token, and chain. A facilitator may verify and broadcast but cannot redirect the transfer. Production requires a remote facilitator; no production buyer or merchant private key enters the gateway.

Consequences. The gateway cannot refund or reverse final settlement. Facilitator failure can stop commerce but cannot redirect funds. The merchant must own reconciliation and customer remediation.

AD-05 - Verification and settlement are separate operations

Context. A verification function that can move funds makes retries and classification unsafe.

Alternatives. One opaque pay() call; separate side-effect-free verification and side-effecting settlement.

Decision. Every payment provider implements createRequirement, verify, settle, and health. verify must never move funds. After a successful verification, the pipeline durably reserves a provider-derived authorization identity before calling settle.

Consequences. Concurrent and post-restart replay are stopped before value moves. Provider contracts are stricter, and a settlement timeout must be represented as uncertain when broadcast may already have occurred.

AD-06 - Configuration-first integration with strict startup validation

Context. The merchant should not rewrite a backend, while a misconfigured price, URL, network, token, or destination can lose money or expose internal systems.

Alternatives. Backend SDK; dynamic administrative API; versioned declarative configuration.

Decision. A strict, versioned YAML document defines the merchant, server, storage, resources, protocol exposure, and payment rails. Environment substitution happens before validation except for the schema-version gate. Unknown keys, unsupported features, unsafe backend URL templates, disabled rails, development destinations on public networks, and incomplete mainnet acknowledgements stop startup.

Consequences. Deployments are reproducible and reviewable, but configuration changes require validation and restart. OpenAPI import generates a draft of this configuration and never publishes operations automatically.

AD-07 - SQLite first, storage interfaces and PostgreSQL migration seam

Context. The core product needs atomic replay reservation and durable receipts without requiring an external database, while larger production scale requires high availability.

Alternatives. In-memory state; PostgreSQL from the first release; embedded SQLite behind a narrow repository.

Decision. The default store uses SQLite WAL with owner-only files and a schema containing receipts, events, and payment attempts. The replay key has a unique constraint. Core logic depends on a storage interface, not SQL. PostgreSQL is a second adapter behind the same interface and is in place before multi-replica operation is supported.

Consequences. Local operation is simple and deterministic. Single-node production depends on filesystem backup. A PostgreSQL migration must preserve uniqueness, transaction semantics, ordering, redaction, and health behavior rather than merely copy tables.

AD-08 - Exact protocol and dependency revisions

Context. Protocol schemas, SDK behavior, EIP-712 domains, and header formats can change incompatibly under nominally compatible package versions.

Alternatives. Floating dependency ranges; vendor all protocols; exact pins with explicit migration decisions.

Decision. Protocol-facing dependencies, toolchains, and CI actions are pinned to reviewed revisions. Optional protocol and payment dependencies are separate peers or entry points so unused rails do not expand every deployment. Runtime descriptors publish supported revision, capabilities, unsupported features, and status.

Consequences. Upgrades require deliberate compatibility work. Operators gain a testable support claim and a smaller default dependency surface.

AD-09 - Synchronous single-shot core; scoped state for session protocols

Context. The canonical commerce model is one request, one price, one settlement, one backend result. ACP is session-shaped, and A2A also defines a larger asynchronous task lifecycle.

Alternatives. Expand the entire core into a workflow engine; emulate sessions in memory; keep session state inside the relevant adapter while one terminal action enters the pipeline.

Decision. The canonical pipeline is synchronous and single-shot. A2A supports discovery and synchronous message/send only. ACP owns checkout-session state through a narrow store sharing the commerce database. complete must acquire a durable idempotency/terminal-state reservation before invoking the pipeline. Receipt creation and terminal session commit must participate in one storage unit of work, or the adapter is not released.

Consequences. Long-running tasks, streaming, subscriptions, and post-order lifecycle remain out of scope. The storage abstraction must expose a bounded transaction-coordination seam before ACP; a design that writes a receipt and then separately closes a session has an unacceptable duplicate-delivery crash window.

AD-10 - AP2 is authorization policy, not transport or settlement

Context. AP2 proves human intent through a signed mandate chain but does not replace the transport or final payment rail.

Alternatives. New AP2 adapter; accept any valid credential without cart binding; pre-settlement authorization policy.

Decision. AP2 mandate verification runs before payment settlement. It verifies SD-JWT VC signature, Key Binding JWT, configured did:key issuer, expiry, audience, nonce/context, disclosed resource, and exact resolved amount. Undisclosed required claims count as missing. The mandate reference is stored as receipt provenance; the mandate itself and signing keys are not.

Consequences. A valid mandate never replaces payment. Network DID resolution is excluded from the supported AP2 slice, keeping verification deterministic and preventing a resolver outage from being mislabeled as buyer rejection.

AD-11 - ACP delegated payment credentials pass through to the merchant

Context. In ACP the merchant remains merchant of record and charges a delegated token using its existing payment provider.

Alternatives. Gateway charges the credential; gateway vaults it for retries; gateway passes it once to the merchant backend.

Decision. The gateway validates the protocol envelope and forwards the delegated token only during complete. It neither persists, logs, displays, nor charges the token. ACP capability negotiation states the supported checkout slice and rejects unsupported carts cleanly.

Consequences. PSP integration and card-data obligations remain primarily with the merchant, but the gateway still handles sensitive credential material transiently and must be reviewed accordingly. Multi-item behavior, session expiry, and supported capabilities are design gates for that release.

AD-12 - Fail-closed errors with honest uncertainty

Context. A buyer error, dependency outage, failed settlement, settlement whose confirmation timed out, and backend failure after settlement are materially different facts.

Alternatives. Generic failure; optimistic delivery; stable typed outcomes.

Decision. The gateway uses stable, client-safe error codes and keeps raw dependency errors in restricted logs. No proof or invalid proof means no delivery. Provider unavailability is retryable and never payer blame. A broadcast settlement that cannot be confirmed is settlement-uncertain, not failed. Backend failure after settlement records paid-but-undelivered state.

Consequences. Clients can respond safely and operators can reconcile. The gateway does not attempt automatic refunds or re-use an authorization whose settlement state is uncertain.

AD-13 - Deterministic tests; models are clients, never dependencies

Context. Core correctness cannot depend on non-deterministic model output or public services

Alternatives. Hosted-agent integration tests; mocks only; deterministic clients plus a private chain and real protocol SDKs.

Decision. Routing, validation, payment, settlement, receipts, and adapters are deterministic. CI uses a programmatic buyer, local Anvil, an EIP-3009-compatible mock token, and real protocol client SDKs for conformance. Public-network and real-fund tests are separately configured, manual, and skipped without credentials.

Consequences. CI can prove balance changes and protocol behavior offline. Model quality and agent planning remain outside the gateway's assurance case.

AD-14 - Replay reservation is mandatory; post-effect audit writes are best-effort

Context. Failure to reserve an authorization before settlement can cause a duplicate charge or delivery and must stop theflow. By contrast, once payment has settled or a backend has returned, failure to write an event or receipt cannot safely undo the external effect.

Alternatives. Treat every persistence call as fatal; ignore all persistence errors; make the pre-settlement reservation fatal and surface later audit gaps without rewriting the business outcome.

Decision.Replay reservation is a mandatory, fail-closed write before settlement. Payment-attempt updates, events, and receipts written after an external effect are attempted, and their failures are logged and counted, but they do not relabel a settled payment or a returned backend result as if it never happened. Readiness fails when storage is known to be unwritable.

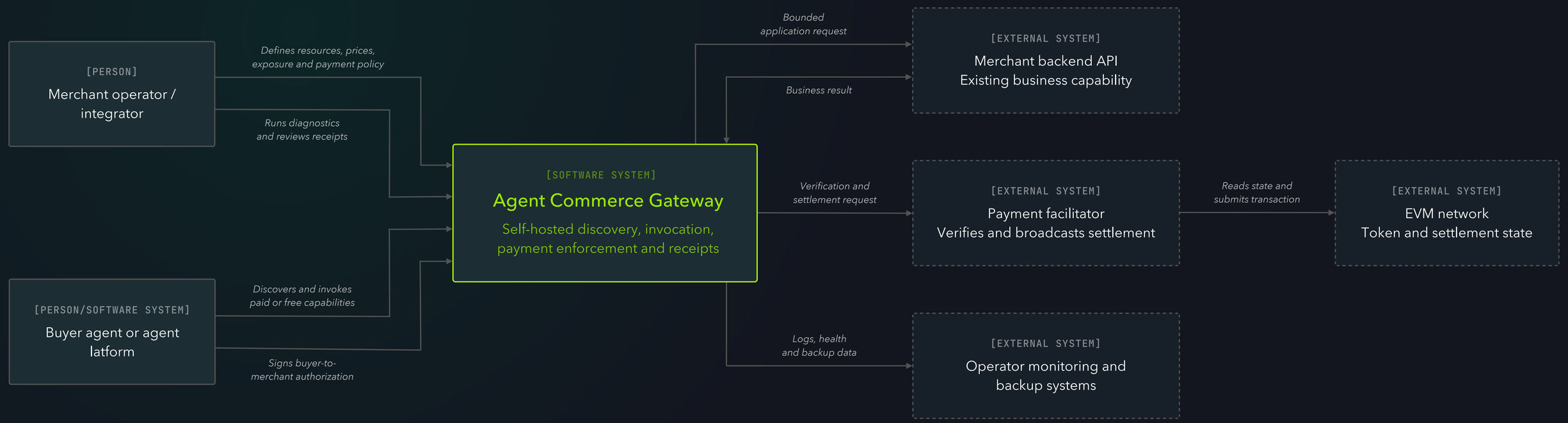
Consequences. The system preserves truthful commerce semantics but can have an audit gap during a storage failure. Production monitoring must alert on the first failed audit write, and reconciliation must use chain and backend evidence. ACP cannot use this best-effort rule for terminal session state: its completion reservation must be durable before execution as specified by AD-09.

Technical architecture - C4

Legend in diagrams: solid green/blue components belong to the initial release; dashed blue components belong to extended scope or stand behind a design gate.

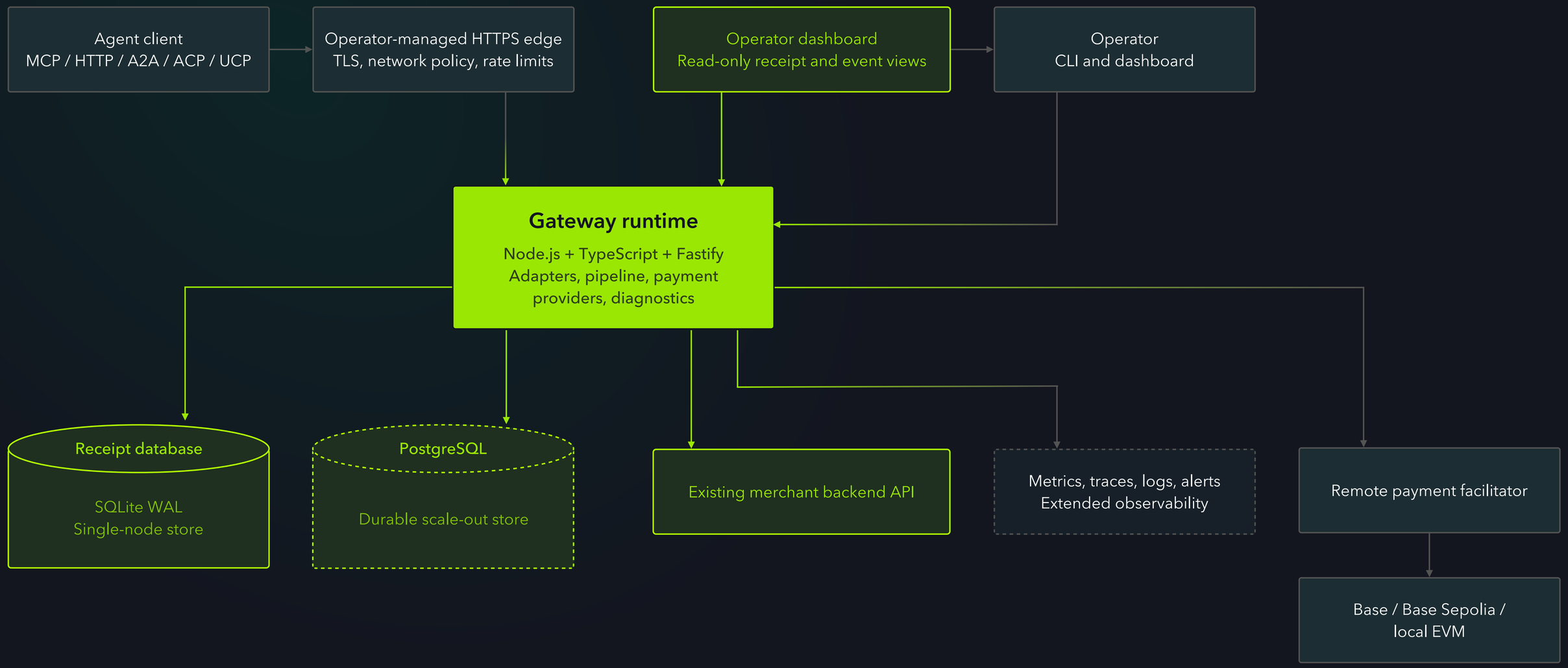
📌 Agent Commerce Gateway

System Context Diagram



📌 Agent Commerce Gateway

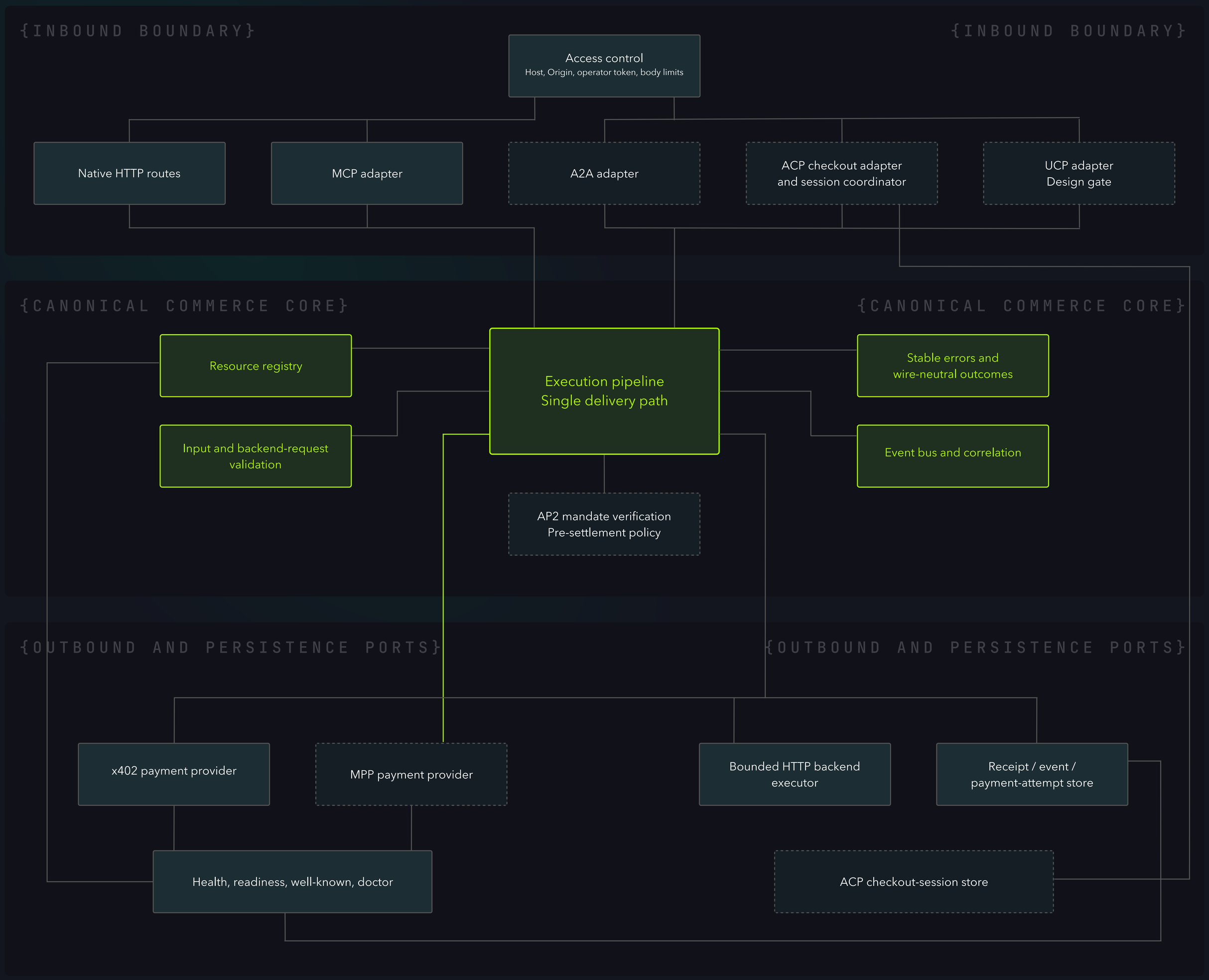
Containers Diagram



SYSTEM	RESPONSIBILITY	DATA OR AUTHORITY
Operator edge and observability	Public discovery and invocation, adapter mounting, canonical pipeline, payment enforcement, backend relay, receipts, health.	Holds configuration and transient proofs; no buyer or merchant production key.
Receipt database	Replay reservation, payment attempts, receipts, events, ACP sessions.	Operational source for gateway audit; not authoritative over chain settlement.
Operator dashboard	Read-only local visualization of health, receipts, and event sequence.	Holds a browser-readable demo token only in the local demo; production access needs a server-side credential pattern.
CLI	Initialize and validate configuration, diagnose dependencies, run deterministic demo.	Reads deployment configuration; must mask secrets.
Merchant backend	Execute the business capability.	Authoritative for delivered body and business-side processing.
Facilitator	Verify/broadcast supported payment authorizations and return settlement reference.	May hold a gas-paying key; cannot redirect a buyer authorization.
EVM network	Token balances, authorization state, and final settlement.	Financial source of truth for x402/MPP EVM paths.
Operator edge and observability	TLS, exposure policy, rate limiting, metrics retention, alerts, logs.	Operator-owned production controls.

📌 Agent Commerce Gateway

Component Diagram



Component contracts

BOUNDARY	REQUIRED CONTRACT
Protocol adapter	Translate wire request to canonical request; call the pipeline once; translate outcome; publish exact capabilities; never invoke backend or settle payment.
Payment provider	Create challenge; verify without moving funds; derive replay key from authorization; settle; distinguish rejection from unavailability; publish health and unsupported features.
Backend executor	Allow only configured HTTP methods and destinations; encode path/query safely; enforce timeout and response-size cap; do not relay backend error bodies.
Receipt store	Atomically reserve replay key; attempt redacted payment-state, receipt, and event writes; bound list queries; expose health; support versioned migrations; distinguish mandatory pre-settlement writes from best-effort post-effect audit writes.
Authorization policy	Verify mandate and resolved-commerce binding before settlement; never issue mandates or hold signing keys.
Diagnostics	Report effective, non-secret configuration and dependency health; anonymous endpoints use a fixed safe vocabulary.

Canonical domain model

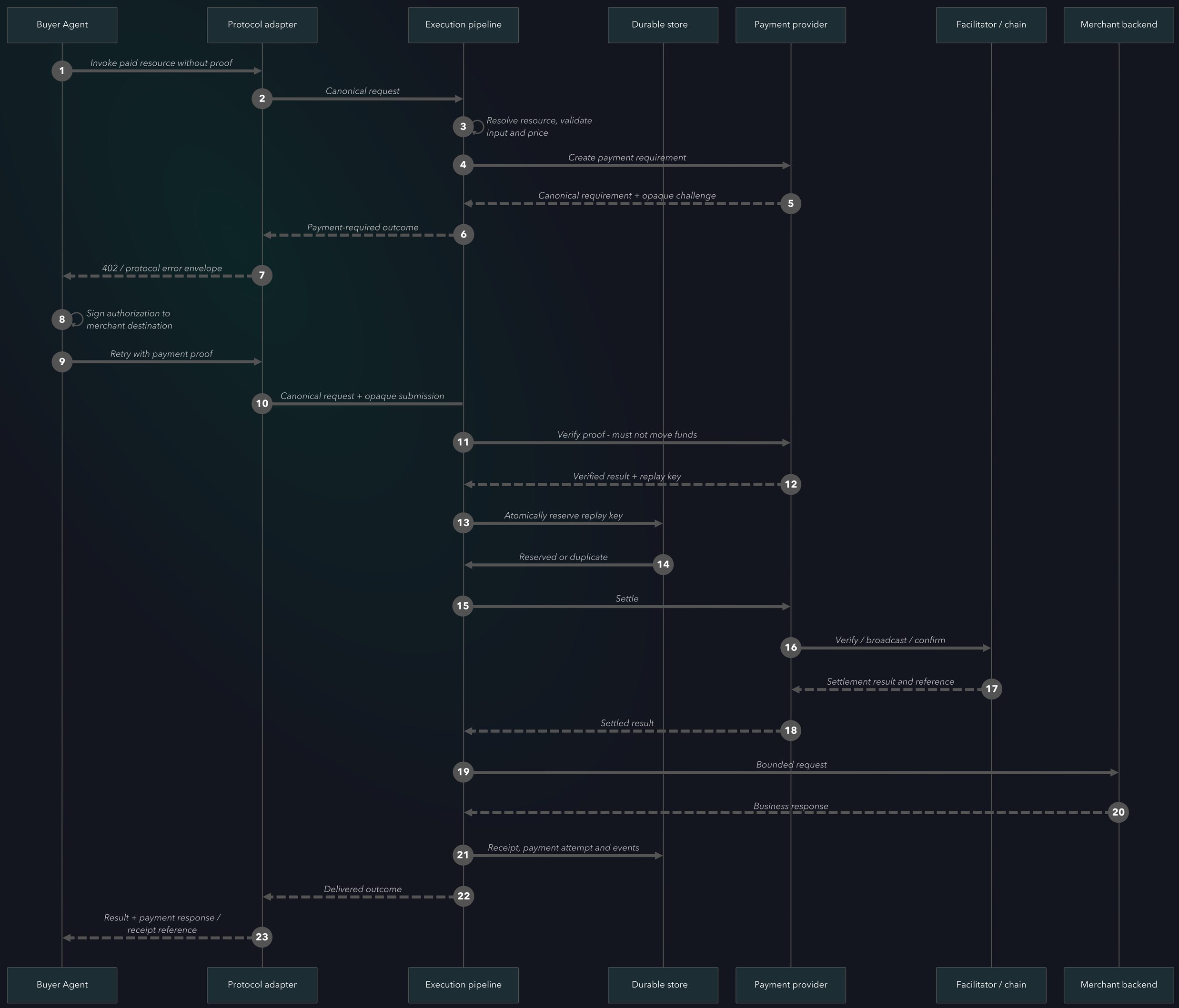
CONCEPT	PURPOSE	KEY INVARIANT
CommerceResource	One merchant capability	Contains protocol exposure, input/output schema, backend handler, price, and allowed rails.
CanonicalRequest	One normalized invocation	Carries one request id, resource id, input, protocol, optional opaque payment proof, and receive time.
PaymentRequirement	Canonical amount and destination plus provider-native challenge	Core may pass opaque challenge data but does not interpret rail-specific fields.
PaymentResult	Verification or settlement outcome	Verified result has a provider-derived replay key; only settled result authorizes backend delivery.
ExecutionOutcome	Delivered or payment-required result	All other failures are stable typed errors.
CommerceReceipt	Evidence of the gateway's outcome	Records correlation, resource, payment summary, backend status, duration, protocol, and non-secret metadata.
CommerceEvent	Observable state transition	Shares request id and never blocks commerce if an optional subscriber fails.
PaymentAttempt	Durable authorization lifecycle	Unique replay key; terminal states distinguish settled, rejected, failed, and uncertain.

CheckoutSession	ACP-owned pre-completion state	Has expiry and one terminal transition; delegated token is never stored.
MandateReference	Provenance for verified AP2 authorization	Stores only the minimum reference and verification result needed for audit.

Canonical monetary amounts are decimal strings in display units. Providers convert them to integer base units without floating point. Provider-native payment payloads, delegated tokens, signatures, secrets, and full backend bodies are never persisted in receipts or events.

📌 Agent Commerce Gateway

Paid Request Sequence



Failure semantics

FAILURE POINT	EXTERNAL OUTCOME	DELIVERY	DURABLE FACT
Missing proof	Payment required	No	Challenge event may be recorded
Malformed, expired, wrong amount/network/asset	Payment invalid	No	Rejection without raw proof
Duplicate authorization	Payment replayed	No	Original unique reservation remains
Provider or verifier unavailable before verdict	Retryable unavailable	No	Infrastructure failure, not payer rejection
Settlement rejected before broadcast	Settlement failed	No	Failed attempt
Broadcast occurred but confirmation timed out	Settlement uncertain	No	Transaction reference retained when known; no automatic retry with same authorization
Backend fails after settlement	Backend error / timeout	No	Settled payment plus paid-but-undelivered event
Persistence cannot reserve replay key	Storage error	No settlement	Fail closed
Receipt/event persistence fails after an external effect	Preserve truthful payment/ backend outcome; operator alert	Payment or backend effect may already exist	Audit gap is logged and counted; readiness fails when storage is known unwritable

Protocol and payment mapping

STANDARD	ARCHITECTURAL ROLE	DELIVERY	DURABLE FACT
Native HTTP	Protocol surface	Resource list and synchronous invoke; payment headers and JSON envelopes	No general proxying or arbitrary routes
MCP	Protocol adapter	Tool discovery and invocation over Streamable HTTP	No resources, prompts, sampling, roots, completion, server requests, or notifications
x402 v2	Payment provider	exact, EVM, EIP-3009, USDC-like token, local/remote facilitator	No Solana, deferred scheme, Permit2, dynamic routing, or multi-asset selection

A2A	Protocol adapter	Agent Card plus JSON-RPC message/send	No gRPC, REST binding, streaming, task lifecycle, push, or multi-turn context
AP2	Authorization policy	SD-JWT VC + Key Binding, did:key allowlist, payment-channel carriage	No issuance, network DID resolution, external trust list, refunds, or custody
ACP	Stateful protocol adapter	Five Checkout API endpoints with capability negotiation	No product feed, discounts, extensions, order lifecycle, refunds, or gateway charging
MPP	Payment provider	Core + charge + EIP-3009 on Base; exact revisions pinned at implementation start	No authorize/subscription, Tempo/Solana, Permit2, card/ACH, streaming, cancellation, or extensions
UCP	Protocol adapter candidate	Design gate; slice not selected	Must not be advertised until revision, slice, state model, and conformance plan are pproved

Data ownership and consistency

DATA	AUTHORITY	KEY INVARIANT
Resource definition and price	Validated merchant configuration	Loaded as one versioned startup snapshot; runtime does not silently mutate it
Backend business result	Merchant backend	Gateway relays a bounded successful result; receipt stores only non-secret summary metadata
Payment settlement	EVM chain / applicable rail	Gateway payment result must carry a verifiable external reference; local receipt cannot override chain state
Replay reservation	Gateway database	Unique provider-derived authorization key is committed before settlement
Receipt and events	Gateway database	Correlated by request id; operational audit, not a financial ledger replacing chain state
ACP session	Gateway dACP adapter storeatabase	Terminal transition and receipt participate in one transaction or equivalent durable idempotency protocol
AP2 trust configuration	Merchant configuration	Empty issuer allowlist means no issuer is trusted; no network resolution in the supported slice

Public and operator surfaces

SURFACE	AUDIENCE	PROTECTION
Health, public descriptor, resource discovery	Agents and operators	Public, coarse and non-secret; host/origin policy still applies
Resource invocation and protocol mounts	Buyer agents	Payment and input validation, not identity authentication
Readiness	Operators and orchestration	Public fixed-vocabulary status; dependency checks memoized/single-flight
Receipts, events, event stream	Merchant operators	Header bearer token with constant-shape comparison; no configured token means route is closed
Dashboard	Local demonstration	Loopback only; production requires a server-side session or proxy and must not embed the operator token in browser JavaScript

Quality attributes and SLOs

The following are acceptance targets and hard architectural limits that a deployment must meet, not observed production performance.

Hard limits and measurable objectives

SCENARIO	METRIC / TARGET	SCOPE AND MEASUREMENT
Oversized inbound request	Maximum body 256 KiB on native and mounted protocol surfaces	Reject or terminate before unbounded parsing; test both Content-Length and chunked bodies
Oversized backend response	Maximum body 1 MiB	Abort read and return a bounded backend error; never buffer beyond cap
Stalled merchant backend	Default timeout 10 s, configurable per resource and always finite	Measured from outbound start to response/abort
MCP parse pressure	At most 20 concurrent mounted HTTP requests per adapter instance	Excess receives retryable busy response before protocol parsing
MCP invocation fan-out	At most 8 executing tool calls and 64 queued tool calls per adapter instance	Excess fails without reaching the merchant backend
Event-stream pressure	At most 32 concurrent authenticated subscribers; heartbeat every 15s	Slow subscribers drop live frames instead of growing an unbounded queue

List query pressure	Default 50, hard maximum 500 rows	Enforced in storage, not only HTTP handlers
Readiness amplification	Dependency result cached 2 s and concurrent probes share one evaluation	A burst creates at most one upstream health evaluation per cache window
Discovery latency	p95 <= 200 ms, p99 <= 500 ms	Gateway response time at edge, excluding client network; 30-day window
Gateway processing overhead	p95 <= 100 ms, p99 <= 250 ms	End-to-end request minus measured backend and settlement dependency time
Free invocation overhead	p95 <= merchant-backend duration + 150 ms	Excludes client-to-edge network; includes validation, persistence, and mapping
Paid invocation overhead after proof	p95 <= settlement duration + backend duration + 300 ms	Separately record verification, reservation, settlement, backend, persistence
Single-node availability	>= 99.5% monthly for discovery and free invocation when declared dependencies are healthy	Excludes planned maintenance; paid flow tracked separately by dependency and rail
PostgreSQL production availability	>= 99.9% monthly for gateway-owned surfaces	Requires at least two gateway instances, external edge, PostgreSQL HA, and tested failover
Payment replay safety	0 duplicate settlements or uplicate deliveries from one authorization	Release-blocking invariant; concurrent and restart tests required
Secret persistence	0 raw payment proofs, delegated payment tokens, private keys, seeds, or bearer credentials in receipt/event rows	Automated negative scans and fixture assertions
Audit-write failure detection	Alert within 1 min of the first failed post-effect receipt/event/payment-attempt update	Does not rewrite the external outcome; opens a reconciliation incident
Recovery - SQLite	RPO <= 24 h, RTO <= 60 min	Encrypted daily backup plus restore rehearsal; operator may choose stricter values
Recovery - PostgreSQL	RPO <= 5 min, RTO <= 30 min	Continuous backup/PITR and documented failover/restore rehearsal
Security incident response	Acknowledge critical alert <= 15m; containment decision <= 60m	Operator-owned on-call SLO; required before public mainnet exposure

Availability objectives do not hide dependency failures. Dashboards split gateway-owned errors, merchant-backend failures, facilitator/RPC unavailability, buyer rejection, settlement failure, settlement uncertainty, and paid-but- undelivered outcomes. No single "success rate" may combine them.

Reliability properties

The gateway is fail closed for delivery, not globally fail stop. Invalid configuration and unavailable durable replay storage stop startup or readiness. An optional protocol adapter may be isolated as unhealthy while other adapters continue, but a configured paid resource is never downgraded to free.

Delivery is not described as "exactly once" in the general case. A settled payment followed by backend timeout is inherently ambiguous to the merchant unless that backend provides its own idempotency contract. The gateway retains paid-but-undelivered state when storage is available, but it cannot manufacture exactly-once behavior for an arbitrary HTTP backend.

An authorization is at-most-once at the gateway because reservation is durable and unique before settlement. On-chain EIP-3009 state is a second, independent double-spend defense.

A post-effect receipt or event write failure does not reverse payment or backend work. It creates a measurable audit gap and requires reconciliation.

SQLite is not a high-availability database. Multi-replica claims are forbidden until PostgreSQL, distributed health, and transaction semantics are qualified.

Observability contract

The gateway emits structured, redacted logs, request ids, receipts, events, durations, health, readiness, and runtime descriptors. Extended observability adds metrics and traces with bounded-cardinality labels:

- request count and duration by route template, adapter, outcome, and resource identifier from configuration;
- payment verification, settlement, rejection, unavailable, uncertain, and replay counters by provider and network;
- backend duration, timeout, non-2xx, and response-too-large counters;
- persistence latency/error, queue depth/rejection, active SSE subscriber, and readiness status gauges;
- paid-but-undelivered count and oldest age;
- no payer address, transaction payload, proof, token, secret, raw URL query, or unbounded caller value in metric labels.

Threat model - STRIDE

Assets and trust boundaries

Protected assets are merchant revenue, buyer authorization, merchant settlement destination, resource pricing and exposure, backend confidentiality and availability, receipt and replay integrity, delegated checkout credentials, AP2 mandate provenance, operator ledger access, facilitator credentials, RPC credentials, and package publication authority.

Trust boundaries exist between agent and gateway edge, browser dashboard and operator routes, gateway and merchant backend, gateway and facilitator, gateway and RPC/chain, gateway and database/filesystem, configuration/environment and runtime, optional protocol dependency and core, and CI/release workflow and npm.

CATEGORY	ATTACK VECTOR	ARCHITECTURAL RESPONSE	RESIDUAL EXPOSURE
Tampering	Change amount, recipient, network, asset, validity, or nonce	Signed EIP-3009/EIP-712 fields plus provider verification against resolved requirement	Token-contract or chain compromise remains external
Tampering	Replay one authorization through another request or adapter	Replay key derived only from authorization identity, unique durable reservation before settlement, on-chain authorization state	Loss or rollback of the gateway database can remove the first defense; chain state remains the second
Tampering	Bypass payment through a new protocol	Adapter receives pipeline, registry, event and diagnostic ports but no backend executor or settlement authority; conformance test traverses real gateway	Malicious code in the same process can violate language-level boundaries; review and tests remain required
Tampering	Point backend template at an internal or attacker-selected host	Backend host is fixed at configuration; only validated path/query/body fields vary; reject host-position templates and URL userinfo	A malicious merchant configuration can still target internal systems; onfiguration is trusted operator input
Tampering	ACP completion replays after crash	Durable terminal reservation and atomic session/receipt unit of work; reject already completed/cancelled/expired state	Feature cannot ship until this coordination is demonstrated under crash tests
Repudiation	Buyer or merchant disputes what happened	Request id across logs/events/attempt/receipt, payment summary, external transaction reference, backend status, duration	Gateway receipt proves observed process outcome, not content truth, human identity, or legal agreement
Repudiation	Operator or database admin alters local receipts	Restrict database files, backups, and access; compare settlement references with chain	The architecture has no WORM store or cryptographic receipt hashchain

Information disclosure	Raw proof, private key, admin token, delegated token, or backend error leaks	Structured redaction, explicit safe error envelopes, no raw proof/token persistence, backend error body not relayed, bounded safe health vocabulary	Nested or newly named secrets can escape a blocklist; call sites must avoid logging untrusted objects
Information disclosure	Browser bundle exposes operator credential	Initial dashboard restricted to loopback; production requires server-side session/proxy; never embed a real operator token in client assets	Production dashboard authentication is a design gate
Information disclosure	Public RPC redential appears in logs or descriptors	Never publish full RPC URL; expose only network and non-secret endpoint origin when needed	Third-party libraries may includeURLs in thrown errors; sanitize before logging
Denial of service	Oversized JSON, MCP batch fan-out, slow SSE reader, unbounded list	Body/response caps, request/tool semaphores, bounded queue, subscriber cap with dropped frames, list cap	No built-in per-agent quota or adaptive edge throttling; the edge owns that
Denial of service	Readiness probes amplify paid RPC calls	Two-second cache and single-flight dependency evaluation	Sustained low-rate probes still create upstream work; edge rate limit required
Denial of service	Merchant backend, facilitator, RPC, or database outage	Timeouts, health/readiness, dependency-specific outcomes, fail-closed delivery, retry guidance	Single-node SQLite and external dependencies remain availability bottlenecks
Elevation of privilege	Development facilitator key used against a public network	Detect known development keys/addresses, local-node identity probe, mainnet-specific startup acknowledgements	Unknown weak keys cannot be detected automatically
Elevation of privilege	Mainnet enabled accidentally	Explicit allow-mainnet acknowledgement, canonical asset/domain checks, remote HTTPS facilitator, non-development destination, separate unauthenticated-facilitator acknowledgement	Correctly acknowledged but economically unsafe configuration remains operator responsibility
Elevation of privilege	Compromised release pipeline publishes malicious package	Least-privilege workflow token, immutable action SHAs, exact tool versions, full CI before tag publish, npm OIDC trusted publishing, clean-consumer install test	SBOM, artifact signing, and independent reproducible-build verification stand outside the initial release

Critical abuse cases

Compromised gateway host. Rotate operator and facilitator credentials, stop traffic at the edge, preserve database and logs, compare payment references with chain, redeploy from a verified artifact, and restore/reconcile receipts. A host attacker can alter delivery and observe proofs in memory, but must not obtain a merchant or buyer production key because those keys are outside the gateway.

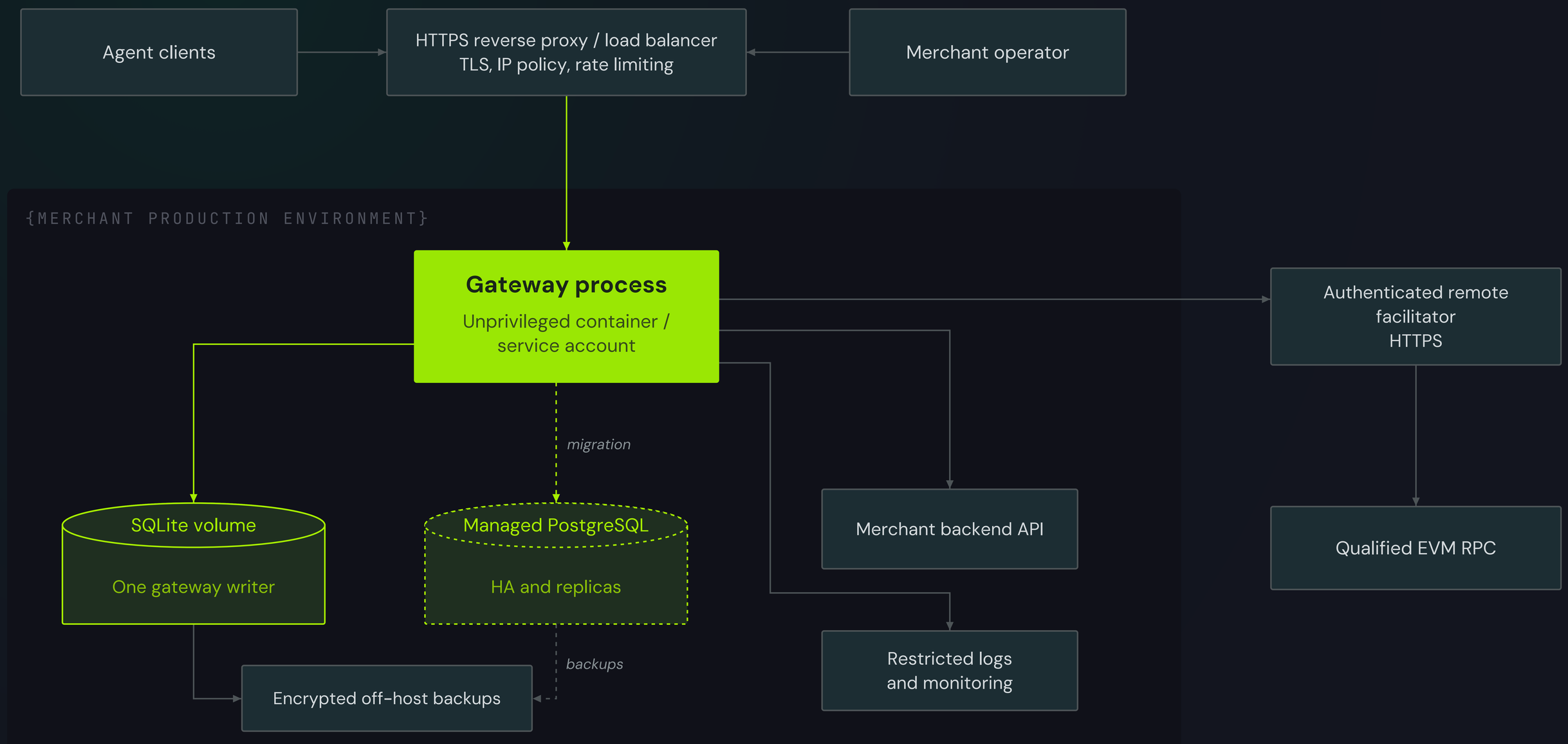
Compromised facilitator. Switch to a qualified facilitator, reject new traffic until health passes, and reconcile every uncertain authorization. The facilitator can censor, delay, or observe authorizations and may spend its gas wallet; it cannot change a signed recipient or amount.

Receipt database loss. Stop paid delivery until a restore or chain-aware reconciliation is complete. On-chain authorization state prevents duplicate spend, but database loss can otherwise permit duplicate backend delivery if an old authorization is presented. Restoring replay state is therefore a security recovery step, not only an audit concern.

Paid but backend failed. Do not automatically retry settlement and do not report payment failure. Retain transaction reference and backend failure, notify the operator, and use merchant policy for re-delivery, refund, or credit.

📌 Agent Commerce Gateway

Deployment Diagram



Single-node production. One gateway process runs as an unprivileged user behind an operator-managed HTTPS edge. SQLite lives on a private durable volume with owner-only permissions and encrypted off-host backup. The gateway uses a remote HTTPS facilitator and a qualified RPC endpoint. Operator routes require a strong token and are separately network-restricted. The demo dashboard is not exposed as a production administrative surface.

Scale-out production. PostgreSQL with backup/PITR replaces SQLite, and at least two stateless gateway instances sit behind the edge. Promotion to this topology requires proof that replay reservation, ACP terminal state, receipt writes, migrations, and readiness behave correctly across concurrent replicas. Running two instances against one SQLite file is forbidden.

Startup and shutdown

Startup order is: load and validate configuration -> open/migrate and test durable store -> build payment providers -> build and isolate protocol adapters -> register routes -> evaluate readiness -> listen -> report effective non-secret settlement destination. An invalid store or paid resource without a provider is fatal. An optional adapter start failure is isolated and visible, not silently advertised as healthy.

Shutdown stops admission at the edge, stops adapters, allows bounded in-flight work to finish where settlement may already have occurred, closes the HTTP server, checkpoints and closes storage, and exits. The production supervisor uses a termination grace period greater than the maximum settlement confirmation window plus backend timeout; a shorter shutdown is permitted only once settlement completion is routed into a durable worker.

CI/CD and release supply chain

The delivery pipeline uses GitHub Actions:

1. Install with the lockfile on Node 22 and pinned toolchains.
2. Check frozen public contracts, format/lint, and type-check the library and dashboard.
3. Run unit, integration, adversarial, protocol-conformance, and coverage tests.
4. Compile and test Solidity contracts with a pinned Foundry release.
5. Run deterministic end-to-end settlement on local Anvil and assert exact balance changes plus transaction reference.
6. Build the npm package and Docker images; install the packed tarball into clean consumers both without and with optional peers.
7. On a version tag, rerun the full CI workflow, dependency audit, and the documented fresh-clone quickstart.
8. Publish to npm only after all gates, using short-lived OIDC trusted publishing; no long-lived npm token is stored in the repository.

Dependency audit findings of high or critical severity require documented human assessment before release. The supply-chain gate adds SBOM creation, artifact checksums and signing, container scanning, and provenance verification.

Backup and disaster recovery

CONCERN	SQLITE MECHANISM	POSTGRESQL MECHANISM
Backup	Consistent SQLite backup API or stopped-copy snapshot; encrypted daily; off host	Continuous WAL/PITR plus encrypted daily full backup
Retention	30 daily and 12 monthly copies unless merchant policy is stricter	Same baseline, with provider retention lock where available
Restore	Restore into isolated path, verify schema/permissions/health, reconcile payment references, then switch	Point-in-time restore to isolated instance, migration check, reconciliation, controlled failover
RPO / RTO	24 h / 60 min acceptance maximum	5 min / 30 min acceptance maximum
Test cadence	Quarterly restore rehearsal and before mainnet launch	Quarterly failover and restore rehearsal
Receipt loss	Stop paid delivery until replay state is restored/reconciled	Same; financial chain state alone is insufficient for duplicate-delivery defense

Configuration, deployment manifests, and secret references are backed up separately. Raw secrets are delivered by the operator's secret manager and are never placed into documentation backups or receipts.

Compliance and data handling

OBJECTIVE	ARCHITECTURAL CONTROL	LIMITATION / OWNER
Non-custody	Buyer authorization pays merchant destination; no merchant/buyer production keys or gateway balances	Legal classification remains deployment- and jurisdiction-specific
Payment credential minimization	Raw proofs and ACP delegated tokens are transient and never persisted or logged	Transient handling still requires security and, where applicable, PCI scope assessment by merchant counsel/assessor
Customer data minimization	Receipt stores resource id, addresses when supplied by rail, amount, status, timing, and transaction reference; no identity dossier	Wallet addresses and purchase history may be personal data; merchant is data controller for retention and access policy

Access control	Operator ledger routes closed by default and protected by token/ network policy	The initial release has one shared operator credential, not RBAC or individual audit identity
Integrity	Signed payment fields, chain reference, replay uniqueness, request correlation	Local audit is database-admin mutable; there is no WORM store or hash chain
Retention and deletion	Configurable operational retention job accompanies the PostgreSQL store; backups follow documented expiry	Chain transactions cannot be erased; legal basis and retention period are merchant responsibilities
Change control	ADRs, pinned revisions, CI tests, tagged release gates, trusted publishing	Independent security audit is required before a high-value production assurance claim
Incident evidence	Restricted structured logs, receipts/ events, dependency outcomes, external references	Operator supplies log retention, immutable sink if required, and on-call process

The gateway does not claim PCI DSS, SOC 2, ISO 27001, GDPR, or another formal certification solely from these controls. A production operator must perform the applicable legal, privacy, security, and payment-scope assessment.

Operational runbooks

Before mainnet, the operator maintains and rehearses runbooks for:

- wrong settlement destination or asset detected before startup;
- facilitator or RPC outage and settlement-uncertain reconciliation;
- paid-but-undelivered customer remediation;
- receipt database corruption/loss and replay-state restore;
- operator token, facilitator credential, and RPC credential rotation;
- malicious or unhealthy adapter isolation;
- rollback to the previous signed package version;
- dependency advisory affecting an optional production rail;
- traffic containment at the edge and evidence preservation after host compromise.

Delivery sequence and architecture gates

Protocol sequence

- **A2A first**. It is the closest proof that discovery and synchronous invocation generalize beyond MCP without changing the core.
- **AP2 second**. Add authorization and receipt provenance on the existing payment channel; the A2A x402 extension is an additional interoperable carriage option, not a replacement for that channel.
- **ACP third**. Introduce protocol-scoped session state only after atomic completion semantics and capability negotiation are designed and crash-tested.
- **MPP fourth**. Add a second rail using the provider interface and existing EIP-3009/replay discipline; pin both MPP Coren and Payment-auth draft revisions when implementation begins.
- **UCP after design approval**. No implementation starts until its supported slice and relationship to resources, checkout, payment, and discovery are written as an ADR.

Cross-cutting requirements

ITEM	ARCHITECTURAL PLACEMENT	GATE
OpenAPI import	Offline/configuration generator	Generated resources require explicit review; no automatic exposure or inferred price/payment policy
Stronger conformance	Real official clients against adapter boundary and full gateway	Every protocol has positive, negative, body-traversal, isolation, and unsupported-capability tests
doctor GitHub Action	CI wrapper over machine-readable diagnostics	No secret output; non-zero status on invalid config or unhealthy required dependency
Shopify/ WooCommerce	Merchant-backend integration examples	No platform-specific type enters core; secrets and webhooks documented safely
PostgreSQL	Receipt/payment/session storage adapter	Unique replay reservation, migrations, transactions, backup, concurrency, and failover tests
Rich observability	Metrics/tracing/logging adapters	Cardinality/privacy review and alerts tied to numerical SLOs

Acceptance matrix

AREA	REQUIRED EVIDENCE BEFORE RELEASE
Canonical boundary	Contract tests prove adapters cannot bypass exposure, validation, payment, replay reservation, or backend executor.
Payment correctness	Wrong amount/recipient/network/asset, malformed, expired, future-valid, insufficient-balance, replay, concurrent replay, restart replay, provider outage, failed settlement, uncertain settlement, and backend-after-payment cases.
Real settlement	Local EVM test proves buyer balance decreases, merchant balance increases by exact amount, and receipt carries real transaction hash.
Protocol honesty	Official client conformance; descriptor revision/capabilities/unsupported list matches served behavior.
Input and SSRF safety	Closed schemas; own-property lookup; encoded path values; fixed backend host; body/response caps; error-body non-relay.
Operator protection	Host and origin tests; token timing/length tests; no-token 404; ledger redaction; dashboard production gate.
Persistence	Permission tests, migrations, bounded lists, failure injection, unique replay races, backup/restore, and ACP transaction crash tests when applicable.
Mainnet safety	Explicit acknowledgement, canonical asset/domain, remote HTTPS facilitator, non-development destination, health capability check, and manual funded smoke evidence.
Package supply chain	Frozen contract, type/lint/tests, clean tarball install, optional-peer isolation, Docker build, dependency review, OIDC publish.
Operations	Numerical dashboards/alerts, restore rehearsal, credential rotation, settlement reconciliation, incident exercise, and rollback.

Decisions required before production mainnet

- Commission an independent security review of payment, replay, backend URL, access-control, redaction, and release paths; resolve all critical/high findings or record formal risk acceptance.
- Select the HTTPS edge, rate limits, network policy, secret manager, log sink, monitoring system, alert routing, facilitator, and RPC providers.
- Approve the SLOs, backup retention, RTO/RPO, privacy retention, incident ownership, and customer-remediation policy for paid-but-undelivered calls.
- Demonstrate restore and replay-state reconciliation, credential rotation, wrong-chain refusal, and a bounded shutdown during an in-flight settlement.

- For any browser dashboard, implement server-side operator authentication; never expose a production operator token in a static browser bundle.
- For multi-replica operation, deliver PostgreSQL and pass concurrent replay, migration, ACP session, failover, and partition tests first.

Readiness rule. A feature may be advertised only when its descriptor, documentation, conformance tests, negative security tests, diagnostics, and operational evidence agree. An unimplemented capability is never reported as partial runtime support.