TECHNICAL OVERVIEW · QODIQA-TO-1.0

The gateway, in engineering *terms*.

Architecture, request lifecycle, cryptographic artifacts, evidence and verification, with every capability labelled Implemented, Specified, Planned or Out of scope.



60 s
MAX ARTIFACT LIFE

364
HTTP OPERATIONS

4
TLA+ MODELS

VERSION 1.0 · 2026-10-29 · FOR ENGINEERS AND ARCHITECTS

QODIQA-TO-1.0 | V1.0 | 2026-10-29

ABOUT THIS DOCUMENT

A technical map of what is built, specified and *not*.

This overview describes the QODIQA enforcement gateway for engineers and architects: how a request is decided and executed, what is signed, what is recorded, who can check it, and where the claims stop.

It is written from the public repositories only: **qodiqa-gateway** (release 1.0.0), **qodiqa-verify** (1.0.0), **qodiqa-anchors**, and the **specification corpus** of 32 documents. Document identifiers such as QODIQA-CORE-2026-001 point to the corpus.

FOUR STATUSES, NEVER MIXED

01 Implemented
In the code and covered by tests or a documented run.

02 Specified
Defined in the standard; not a claim about the gateway.

03 Planned
Listed on the gateway roadmap. Not supported today.

04 Out of scope
Stated as not provided in the scope and limitations page.

WHERE TO START

INTEGRATORS	The path of one request · SDKs and integration
SECURITY REVIEW	Trust boundaries · Threat model summary
VERIFIERS, AUDITORS	The evidence chain · Independent verification
ARCHITECTS	System architecture · Deployment options

DOCUMENT CONTROL

DOCUMENT ID	QODIQA-TO-1.0
TITLE	QODIQA Technical Overview
VERSION	1.0
DATE	2026-10-29
LANGUAGE	English
LICENCE	CC BY 4.0
CONTACT	www.qodiqa.com · contact@qodiqa.com

LICENCES OF THE MATERIAL DESCRIBED

GATEWAY	Apache License 2.0
VERIFIERS, ANCHORS	Apache License 2.0
SPECIFICATION	CC BY 4.0

Boundaries of this document

Technical description only. It is not a certification, an audit opinion or legal advice. QODIQA supports evidence for regulatory frameworks; it does not make anyone compliant. No endorsement or partnership with any named organisation, product or standard is stated or implied.

CONTENTS

Three parts, one *request.*

How a request is decided, how it is proven, and how it is run, verified and read against the standard.

01 How a request is decided 04

1.1 System architecture 05
1.2 The path of one request 06
1.3 The execution lifecycle 07
1.4 Consent and policy enforcement 08
1.5 Semantic inspection, execution and routing 09

02 How it is proven 10

2.1 Cryptographic artifacts and key custody 11
2.2 The evidence chain 12
2.3 Evidence records and anchoring 13
2.4 Identity, tenancy and agents 14
2.5 Trust boundaries 15
2.6 Threat model summary 16

03 Run, verify, reference 17

3.1 Deployment options 18
3.2 SDKs and integration model 19
3.3 Independent verification 20
3.4 Capabilities by status 21
3.5 Glossary 22
3.6 Verify this document 23

PART 01

01

How a request is decided

Architecture, the execution lifecycle, consent and policy, inspection, and execution.

1.1 System architecture 05
1.2 The path of one request 06
1.3 The execution lifecycle 07
1.4 Consent and policy enforcement 08
1.5 Semantic inspection, execution and routing 09

1.1 SYSTEM ARCHITECTURE

One invariant, four moving *parts.*

No AI execution without a valid, non-replayed, policy-permitted authorization artifact. The provider call layer asserts this a second time, independently.

01

Gateway process

An Express application on Node.js 22. `/sign` mints artifacts and `/ai/chat` enforces and executes; operator, evidence and key-discovery endpoints share the process. One writer per audit chain.

02

Redis

Replay nonces, anomaly windows, rate-limit counters and the consent cache. AOF persistence is required in production. If a store is unavailable the request is blocked.

03

PostgreSQL

Required in production. Holds the control plane (policies, key timeline, kill switch), tenancy and RBAC, and the durable evidence stores. Row-level security is depth, not the boundary.

04

Files

The Ed25519 key store, and the NDJSON audit chain with its write-ahead log and checksum, each write fsynced. In production the key store must live outside the application directory.

LAYERING

- API, then middleware, services, core, infrastructure; reverse dependencies fail CI.
- The HTTP contract is generated from the code (364 operations) and diff-checked in CI.
- Four-eyes policy governance and a global kill switch sit on the admin plane.

FAIL CLOSED

- Nonce or anomaly store unavailable: the request is blocked.
- PostgreSQL stopped: 503 without calling the provider; the gateway recovers on its own.
- Beyond its limits the gateway refuses new work (503) instead of queueing without bound.

WHERE THE STANDARD SITS

- The gateway is software. The standard is a separate set of 32 documents under CC BY 4.0.
- Start with QODIQA-CORE-2026-001, QODIQA-ARCH-2026-001 (reference architecture) and QODIQA-SBTM-2026-001 (system boundary and trust model).

qodiqa-gateway

Apache-2.0, 1.0.0. Gateway, SDKs, deployment assets, OpenAPI, protocol vectors.

qodiqa-verify

Apache-2.0, 1.0.0. Offline verifiers in JavaScript and Python, with example sessions.

qodiqa-anchors

Apache-2.0. Append-only checkpoint anchors and the public keys that signed them.

qodiqa

CC BY 4.0. The specification corpus of 32 documents.

1.2 DIAGRAM 1 · REQUEST PATH

The path of one *request.*

01 · SIGN · POST /SIGN
The client authenticates with a per-identity API key. The gateway returns an Ed25519 artifact: at most 60 s, bound to method, path and body, one tenant.

02 · CALL · POST /AI/CHAT
The client sends the request with the header x-qodiqa-artifact. The checks below run in order inside the gateway; the first failure is a BLOCK.

01 Input guards
Content type, body shape, prompt up to 32,768 bytes, artifact header up to 8 KB

02 Verify artifact
Ed25519 signature first, then clock skew, expiry, age and key lifecycle

03 Bind request
Recompute the hash of method, path and body; constant-time compare

04 Consume nonce
Atomic single use in Redis; a replay is a BLOCK, and so is a store failure

05 Anomaly windows
100 per key and 5,000 global events per 60 s; a store failure blocks

06 Consent + policy
Rules in priority order; the first refusal decides; ten types decide

07 Inspect
Semantic tiers ALLOW to DENY, plus DLP and content rules where configured

08 INTENT entry
Written with fsync before the provider is called; a refusal has none

09 Provider call
Execution invariant checked, budget charged, routing, one failover hop

10 Output check
The answer is inspected before it is returned or, when streaming, released

11 Decision record
Audit entry (hash chain) and transparency leaf; an ALLOW that cannot be recorded fails

12 Response + receipt
The decision and response; a signed receipt when both entries were stored

BLOCK PATH Any refusal is a BLOCK with a reason code (HTTP 200 by default). Refusals after step 02 are audited; steps 01 and 02 are counted and sampled.

Source: enforcement flow and protocol sections 4 to 6. When the kill switch is active, /sign and /ai/chat answer 503 before any of this runs.

1.3 THE EXECUTION LIFECYCLE

Ordered checks, first failure *wins.*

STEP	CHECK	FAILURE REASON	AUDIT
00	Kill switch: while active, /sign and /ai/chat answer 503 before anything else runs	KILL_SWITCH_ACTIVE; state not confirmable: KILL_SWITCH_STATE_UNAVAILABLE	Not audited
01	Input guards and artifact decode: JSON body, prompt up to 32,768 bytes, header up to 8,192 bytes, base64 JSON	MISSING_PROMPT, PROMPT_TOO_LARGE, MISSING_ARTIFACT, ARTIFACT_TOO_LARGE (413), ARTIFACT_DECODE_FAILED	Counted, sampled
02	Verify artifact: exactly five keys; Ed25519 signature before any clock or key check; then 5 s future skew, expiry, 60 s age, key lifecycle	INVALID_ARTIFACT, INVALID_SIGNATURE (also for an unknown key), EXPIRED, TIME_SKEW_EXCEEDED, KEY_REVOKED	Counted, sampled
03	Bind request: recompute the hash of method, path and body and compare in constant time	REQUEST_MISMATCH	Audited
04	Nonce: per-artifact revocation check, then atomic single-use registration in Redis, with a restart barrier	ARTIFACT_REVOKED, REPLAY_DETECTED, REPLAY_STATE_UNPROVABLE, NONCE_STORE_ERROR	Audited
05	Anomaly: sliding windows of 100 events per key and 5,000 global per 60 s	ANOMALY_DETECTED (SIGNATURE_BURST, RATE_BURST, STORE_ERROR)	Audited
06	Consent and policy: enabled rules in priority order; the first refusal decides	CONSENT_REVOKED, IP_BLOCKED, OUTSIDE_TIME_WINDOW, RATE_LIMIT_EXCEEDED, or an ABAC reason	Audited
07	Inspect: semantic tiers; DLP, content rules and budget where configured	Stage SEMANTIC_POLICY; CONTENT_RULE_BLOCKED; BUDGET_EXCEEDED_REQUESTS	Audited
08	Intent and dispatch: INTENT entry written with fsync before the provider call; execution invariant asserted	UNVERIFIED_EXECUTION_ATTEMPT; PROVIDER_TIMEOUT, PROVIDER_UNAVAILABLE	Audited
09	Record: DECISION entry and transparency leaf; for an ALLOW the write is mandatory	AUDIT_PERSISTENCE_FAILED, TRANSPARENCY_PERSISTENCE_FAILED (503); the provider answer is discarded	The record

Signature before clock

A forged artifact always receives INVALID_SIGNATURE, so it learns nothing about the clock or which keys exist.

Nonce after binding

Only a correctly signed, correctly bound artifact consumes a nonce, so a malformed request cannot burn a valid requestId.

Anomaly after nonce

Replays do not use anomaly quota: only novel requests increment the windows. BLOCK is HTTP 200 by default; QODIQA_BLOCK_HTTP_MODE=status changes it.

1.4 CONSENT AND POLICY ENFORCEMENT

Eleven rule types, ten *decide*.

RULE TYPE	WHAT IT ENFORCES	APPLIED BY
REQUIRE_CONSENT	An active consent for the caller, provider and scope; pending, suspended, revoked or expired consent blocks	Engine loop
BLOCK_IP	Exact addresses and CIDR ranges	Engine loop
TIME_WINDOW	Execution only inside configured UTC time-of-day windows	Engine loop
RATE_LIMIT	Per-key sliding window in Redis; a store error counts as exceeded	Engine loop
ABAC	A JSON condition tree (DENY or REQUIRE) over subject, tenant, request, resource, environment, tool and agent attributes	Engine loop
DATA_RESIDENCY	The chosen provider must map to a region the operator allows; an undeclared region is refused	Engine loop
DLP	Detect, redact or block personal data and secrets, by direction: input, output, tool input, tool output	Dedicated stage
CONTENT_RULE	Tenant words, phrases and linear-time regular expressions over requests and answers	Dedicated stage
BUDGET	Limits on requests, tokens and cost; charged just before the provider call, settled afterwards	Dedicated stage
ROUTING	Operator-listed candidate providers and models, chosen by a strategy	Dedicated stage
DATA_RETENTION	Declares retention days per data category; validated and hashed, always passes	Declaration only

Order, governance and drift

Sorted by priority, hashed into policyHash; three operators can be required. Drift only warns.

Consent

Five states. Revocation blocks the next request; across instances a stale ALLOW lasts at most 60 s.

1.5 SEMANTIC INSPECTION, EXECUTION AND ROUTING

Inspection is a *layer,* not the guarantee.

01
Normalise
Unicode folding, invisible characters stripped, base64, URL and HTML escapes decoded, look-alike letters folded.

02
Classify
Thirteen classifiers, each a pure function of the text.

03
Score
Confidence times a severity weight; the worst finding wins.

04
Act
The tier blocks or allows; evidence (decision, score, matched classifiers) joins the audit entry, never the raw text.

SEMANTIC INSPECTION

- **13 bundled rule classifiers:** jailbreak, prompt injection, credential and system-prompt extraction, tool abuse, policy bypass, data exfiltration and others.
- **Tiers at default thresholds:** ALLOW below 0.35, WARN from 0.35, REVIEW from 0.55, DENY from 0.70. DENY always blocks; REVIEW blocks unless opted out; WARN blocks only if opted in.
- **Deterministic:** no classifier calls a model. The same checks run on output, and on streaming and non-streaming paths.
- **Embedding layer:** local ONNX inference exists in the code, but its model and reference resources are not distributed in the public repository. Without them the gateway starts only with `QODIQA_SEM_EMBEDDING_ENABLED=false` and screening is limited to the rule engine.

EXECUTION AND ROUTING

- **Second guard:** a verified context is asserted before every provider dispatch (UNVERIFIED_EXECUTION_ATTEMPT otherwise). It does not defend against modified gateway code.
- **Eight adapters:** openai, anthropic, gemini, mistral, llama, local, and the experimental azure and bedrock. Only a local OpenAI-compatible server such as Ollama has been exercised against a real server; hosted adapters used simulated transport.
- **Router:** re-checks that the provider is enabled and makes at most one failover hop. Operators can register OpenAI-compatible endpoints at runtime.
- **Streaming:** output is released after each window is checked, in bursts of up to about 400 characters. Released text cannot be recalled, and part of a response can reach the client before the final decision record exists.

Measured limits
Misses paraphrases. Out-of-sample tests found non-English harmful requests undetected (0 of 150 per language); pooled recall of the semantic layer was near 18%, false positives about 15%.

Position in the standard
The Core Standard excludes content inspection, semantic classification and probabilistic risk scoring from the enforcement decision (QODIQA-CORE-2026-001, section 1.6).

Provider output
A provider response is untrusted content and is never executed. Hosted providers use fixed HTTPS destinations; local and llama endpoints are operator-configured.

PART 02

02

How it is proven

Cryptographic artifacts, evidence, identity and tenancy, agents and tools, and the trust boundaries.

2.1 Cryptographic artifacts and key custody 11
2.2 The evidence chain 12
2.3 Evidence records and anchoring 13
2.4 Identity, tenancy and agents 14
2.5 Trust boundaries 15
2.6 Threat model summary 16

2.1 CRYPTOGRAPHIC ARTIFACTS AND KEY CUSTODY

Ed25519, *domain-separated,* canonical.

01

Canonicalise

Keys sorted at every depth, no whitespace, ECMAScript number form. Close to RFC 8785, not identical.

02

Domain-hash

SHA-256 over the tag bytes followed by the canonical bytes. Artifact tag QODIQA::ARTIFACT::V1; request tag QODIQA::REQUEST::V1.

03

Sign

Ed25519 over the raw 32-byte digest, as 88-character base64. There is no algorithm negotiation.

04

Verify

Signature first, so a forged artifact learns nothing about the clock; an unknown key looks like a wrong signature.

THE ARTIFACT

- Exactly five fields: expiry, payload, publicKeyId, signature, timestamp.
- The payload carries requestId (UUID v4), requestHash, policyHash, tenantId, callerContext and the provider.
- Lifetime at most 60 s; a longer requested TTL is clamped and the response says so.
- Protocol V2 also binds the query string and chosen headers; V2 issuance is off by default.

DOMAIN SEPARATION

- 33 registered tags, one per artifact class: artifact, request, audit, policy, receipt, checkpoint, delegation, action, approval, witness cosignature and others.
- A signature for one class is not valid for another.
- Some hashes carry no tag (transparency payload hash, evidence package hash, key fingerprints); Merkle nodes use RFC 6962 byte prefixes.
- QODIQA::AUDIT::V1 is reused for three inputs; recorded as a known deviation.

KEY CUSTODY

- Private keys are encrypted at rest with AES-256-GCM under a key-encryption key (KEK) from the environment or an external KMS.
- Key states: active, deprecated (verifies older artifacts for 65 s), archived, revoked. Emergency rotation revokes the old key at once.
- Vault Transit, HSM, AWS KMS, Google Cloud KMS and Azure Key Vault are exercised against mocks and emulators only.
- Revoking a key in QODIQA does not make an external KMS stop signing; that step is the operator's.

PRIMITIVES IN USE

ED25519 ONLY · SHA-256 · AES-256-GCM · RFC 6962 MERKLE TREES · HMAC-SHA256 EXECUTION INVARIANT · TOTP FOR OPERATOR STEP-UP · CONSTANT-TIME COMPARISONS

2.2 DIAGRAM 2 · EVIDENCE CHAIN

From one decision to a checkable *proof.*

AUDIT CHAIN · APPEND-ONLY, EACH ENTRY HASHES THE ONE BEFORE

ENTRY N-1 — DECISION

prevHash: hash of entry n−2
hash: domain-separated SHA-256 of the canonical fields

ENTRY N — INTENT

Written before the provider call. Linked to its DECISION by the hashed requestId

ENTRY N+1 — DECISION

Outcome, policy and request hash, outputHash of what was delivered, executingProvider

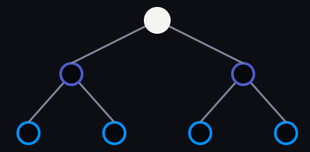
ENTRY N+2 — NEXT ENTRY

prevHash: hash of entry n+1
A change to any earlier field breaks every hash after it

LEAF

HEAD HASH

TRANSPARENCY LOG



RFC 6962 Merkle tree over the entry hashes. Ed25519-signed roots, every 60 s by default. Inclusion and consistency proofs.

SIGNED CHECKPOINT

Chain length, head hash, previous checkpoint hash and key id, signed with a checkpoint key. Defaults: every 1,000 entries or every 15 minutes.

COSIGN

WITNESSES + ANCHORS

Witnesses cosign with their own key and log, and refuse rollback and equivocation. The anchor export is append-only; qodiqa-anchors is a public git repo.

RECEIPT

Signed with a transparency key. Proves the decision was recorded, not that it was correct.

PROOF · QODIQA-PROOF/1

Artifact, audit entries, chain path, checkpoint, inclusion proof, receipt, anchors and public keys, from GET /v1/proofs/:requestId.

OFFLINE VERIFIER

qodiqa-verify: JavaScript and Python, no gateway code.

2.3 EVIDENCE: AUDIT CHAIN, TRANSPARENCY LOG, RECEIPTS, CHECKPOINTS

Four records, each checkable *alone.*

AUDIT CHAIN

- Every decision after artifact verification, EXECUTE or BLOCK, is an entry whose hash covers the previous one.
- Schema v7 is current; v1 to v8 verify. v8 adds agent-action fields, a COMMIT entry and kill-switch records and is off by default.
- Entries hold hashes of requests and outputs, never the text.

TRANSPARENCY LOG

- Each audited decision is a signed leaf in an RFC 6962 Merkle log.
- Signed roots every 60 s by default when PostgreSQL is enabled; inclusion and consistency proofs.
- Public entries never include the payload itself.

RECEIPTS AND PROOFS

- A receipt is signed with a transparency key and returned when both entries were stored; kept 3,600 s by default.
- A proof, qodiqa-proof/1, bundles the artifact, entries, checkpoint, inclusion proof, receipt and anchors.
- A tenant-scoped read withholds the chain path, so the verdict is VALID_PARTIAL.

CHECKPOINTS AND ANCHORS

- A signed checkpoint every 1,000 entries or 15 minutes by default.
- Witnesses (local or exec) cosign; the anchor export is append-only on storage the operator does not control alone.
- A production boot is refused without an anchor export, a witness and witness keys.

A receipt is not a verdict

It proves the gateway recorded the decision, not that the decision was right, and not that the model's answer was correct.

Unanchored means operator-attested

Chain, checkpoints and keys live on the gateway host. A host insider can rewrite and re-sign unless an independent witness or anchor holds the head.

Hashes can confirm guesses

Audit records are permanent. Anyone who can guess a short prompt can confirm it by hashing the guess.

ATTEMPT ON AN ANCHORED DEPLOYMENT	REPORTED BY THE VERIFIER AS	BECAUSE
Truncate the tail of the log	CHAIN_ROLLED_BACK	The log is shorter than an anchored position
Rewrite a prefix and re-chain it	CHAIN_REWRITTEN	The entry at an anchored position has a different hash
Re-sign a checkpoint, even under the real key	CHECKPOINT_REPLACED	Same checkpoint id, different payload
Delete a checkpoint	CHECKPOINT_DELETED	An anchored checkpoint vanished from the live set

2.4 IDENTITY, TENANCY, AGENTS, TOOLS AND MCP

Who calls, and what an agent may *do*.

IDENTITY AND TENANCY

- **Callers:** an individually issued, revocable sk_ signing key per identity; no shared secret. **Operators:** per-operator RBAC keys, with TOTP step-up on changes.
- **Single sign-on:** OIDC tokens, SAML 2.0 and LDAP. A verified token grants nothing until linked to a signing identity. Validated against test infrastructure only.
- **Tenants** are derived from the signing key; keys, nonces, policy sets and consent are per tenant. Organisations own tenants, and a suspended organisation blocks all of them.
- **Isolation** is enforced in the application first. Row-level security adds depth but is not a boundary: the tenant context is a session setting the application role can change.

AGENTS, TOOLS AND MCP

- **Agents and sessions** have gateway-issued ids and limits; the effective limit is the lowest of those requested, the agent's and the platform's. A session can be killed.
- **One action artifact per action:** hashed from the tool call, signed, 60 s, consumed once and in order. **Delegation** is a signed chain of up to 8 links that can only narrow authority.
- **Approval:** irreversible tools need a one-use human approval bound to the exact action, with MFA step-up; the approver is never the requester.
- **MCP tool gateway:** registered servers only, HTTPS, key-pinned, no redirects. Results are untrusted data, screened and delimited. No stdio, no streaming, no server-initiated requests.

Global kill switch

Stops all signing and execution; every replica follows within 5 s, and a replica that cannot confirm the state refuses requests.

Per-artifact revocation

A single artifact can be withdrawn before use; the revocation set holds entries for 120 s, longer than the maximum artifact age.

Tool receipt

A tool-call receipt is a ledger record bound to the delivered output hash. It is not signed.

APPROVAL FLOW FOR AN IRREVERSIBLE TOOL

01
Request
authorize returns 202 pending; the action's place in the session chain is fixed.

→

02
Decide
An operator approves or denies with MFA step-up; never the requester.

→

03
Ask again
The requester repeats the action with the approvalId; the artifact carries approvalRef.

→

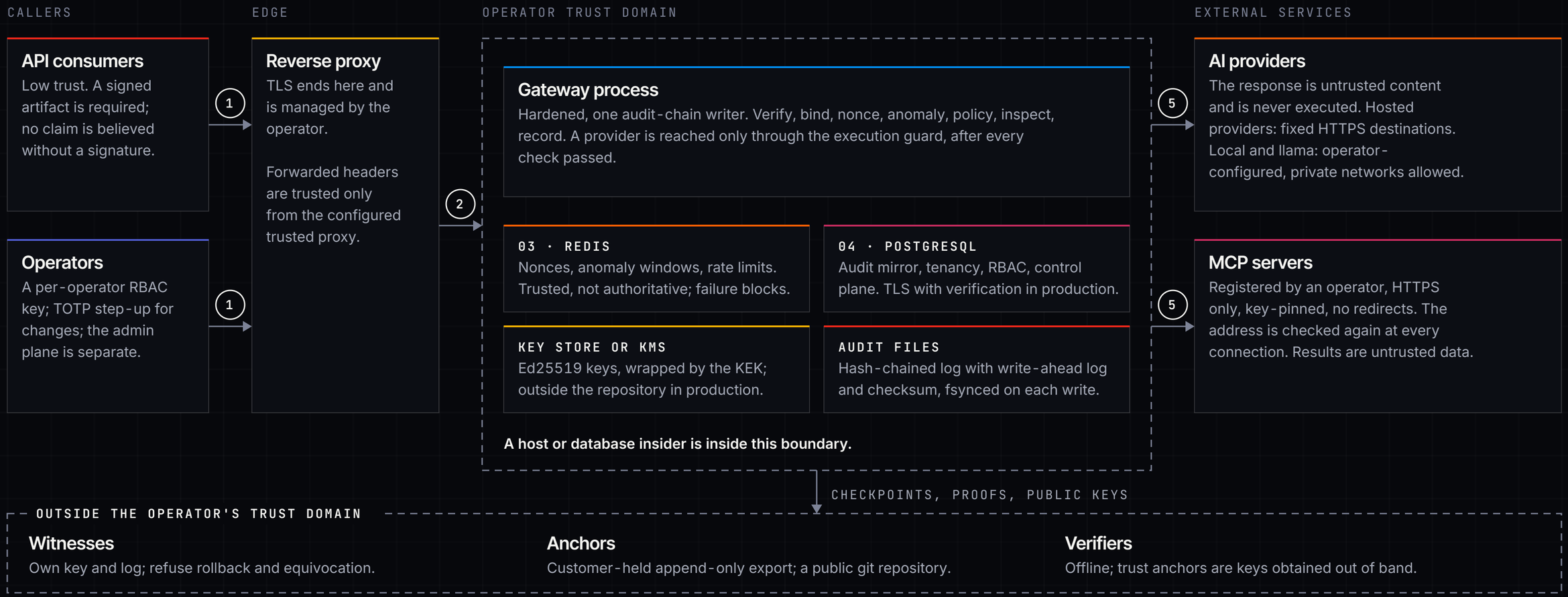
04
Execute
The executor re-checks: still approved, spent once, for this exact action.

→

05
Record
Under audit schema v8 the chain records the request, the decision, INTENT and COMMIT.

2.5 DIAGRAM 3 · TRUST BOUNDARY

Where trust *stops.*



2.6 THREAT MODEL SUMMARY

Threats, controls, and what *remains.*

THREAT	CONTROL IN THE GATEWAY	STATED LIMIT
Forged, altered or replayed artifact	Ed25519 over the full envelope; binding to method, path and body; atomic single-use nonce; 60 s lifetime	Replay protection needs durable Redis (AOF). V1 does not bind headers or the query string.
Spoofed client address	Forwarded headers are honoured only from the configured trusted proxy	Assumes the proxy strips client-supplied forwarded headers.
State store failure	Nonce or anomaly store down: block. PostgreSQL stopped: 503 with no provider call	Availability is traded for safety by design.
SSRF through provider or MCP URLs	HTTPS only; IP literals and private names refused; resolved address re-checked; no redirects	Local and llama providers are operator-configured and may use private networks.
The operator rewrites the record	Hash chain, signed checkpoints, witnesses, anchors and offline verifiers	Without an independent witness the evidence is operator-attested; a local witness is as independent as its custody.
Cross-tenant access	Per-tenant keys, nonces, policy and consent; tenant taken from the signing key	Row-level security is not a boundary on its own.
Injection through tool results	Results are untrusted data: screened, delimited, provenance-marked, bounded in time and size	Screening is heuristic, not a guarantee.
Compromised signing key	Rotation and revocation; keys encrypted at rest; KMS and HSM options	External KMS revocation is the operator's step; integrations tested against mocks and emulators.

ASSUMED, NOT ENFORCED BY THE GATEWAY

A trusted proxy
The load balancer is trusted to strip client-supplied forwarded headers.

A private Redis
Redis sits on a private network; the gateway treats it as trusted but not authoritative.

Synchronised clocks
NTP is assumed. Artifact checks depend on accurate time within the 60 s window.

A safe KEK
Every stored private key depends on the key-encryption key, or on the KMS in its place.

Spec pointers: QODIQA-TM-2026-001 (extended adversarial threat model), QODIQA-SEC-TM-2026-001 (threat model and abuse cases), QODIQA-RISK-2026-001 (residual risk).

PART 03

03

Run, verify, reference

Deployment options, SDKs and integration, independent verification, capability status, glossary.

3.1 Deployment options 18
3.2 SDKs and integration model 19
3.3 Independent verification 20
3.4 Capabilities by status 21
3.5 Glossary 22
3.6 Verify this document 23

3.1 DIAGRAM 4 · DEPLOYMENT OPTIONS

Four shapes, one is *supported* in production.

A · LOCAL SANDBOX

Evaluate in minutes

EVALUATION ONLY

- Gateway with a built-in stand-in provider that echoes the prompt
- Disposable PostgreSQL and Redis
- Semantic screening switched off

`docker compose -f docker-compose.quickstart.yml up`

B · SINGLE NODE

The supported production shape

VALIDATED: ONE INSTANCE

- Reverse proxy ending TLS
- One gateway, persistent keystore and audit volumes
- PostgreSQL (TLS) and Redis (TLS, AOF)

Kustomize single-node overlay, Helm production profile, Compose, systemd, Terraform module.

C · SEVERAL INSTANCES

Each with its own audit chain

NON-PRODUCTION ONLY

- Load balancer routing on /ready
- Instances A and B, separate audit.log each
- Shared Redis, one PostgreSQL

Two instances sharing one audit chain is refused at boot.

D · DISTRIBUTED MODE

PostgreSQL orders one chain

REFUSED IN PRODUCTION

- N replicas, distributed deployment mode
- PostgreSQL is the ordering authority
- No signed checkpoints or anchors yet

Append throughput does not scale with replicas; PostgreSQL becomes a hard dependency of the hot path.

IN EVERY SHAPE

PostgreSQL and Redis are required in production; keys live outside the repository; direct access to providers must be closed at the network level.

Kubernetes and Helm were exercised on local development clusters. The production profile (TLS, shared keystore volume, network policy) needs your own validation.

3.2 SDKS AND INTEGRATION MODEL

Integrate over HTTP, verify *offline.*

01 Provision
One signing identity per caller. The sandbox prints a throwaway key.

02 Sign
POST /sign with the exact request: method, path and body.

03 Send
Header x-qodiqa-artifact: the base64 of the artifact JSON.

04 Handle
Read the decision, EXECUTE or BLOCK with a reason.

05 Keep
Store the requestId; fetch the receipt or proof, verify offline.

CLIENT	REQUIREMENTS	STATUS
JavaScript / TypeScript	@qodiqa/sdk; HTTP client; verifies artifacts and inclusion proofs offline; signing stays on the server	Not published to npm; install from the repository
Python	qodiqa; Python 3.9+, httpx, cryptography	Beta; mostly tested against a mock transport; not on PyPI
Go	Go 1.21+, standard library only; offline artifact verification	Tested internally; not published; protocol V1 only
Java	Java 17+, one dependency (Jackson)	Tested internally; not published; protocol V1 only
Artifact guard	Express and Connect, FastAPI and ASGI, Django	V1 binding; consumes no nonce unless you add a replay hook

API CONVENTIONS

- Core routes also answer under /v1; the OpenAPI file lists 364 operations.
- POST /verify is a dry run: same checks, no nonce consumed, no audit record, no provider call.

CLIENT BEHAVIOUR

- Go and Java clients retry idempotent verbs on 429 and 5xx, and POST on 429 only, never on /ai/chat.
- Go and Java call any documented operation from a table generated from the OpenAPI file.

INTEGRATION CONSTRAINT

- Enforcement happens at the gateway. Direct access to AI providers must be closed at the network level.

3.3 WHAT IS VERIFIED INDEPENDENTLY, AND HOW

Check the record without trusting the *operator.*

LAYER	WHAT IS CHECKED	HOW	LIMIT
Protocol vectors	Canonicalization, hashes, signatures, verification cases, audit chains, transparency	Generated from production code; re-derived by independent JavaScript and Python verifiers that import none of it; drift fails CI	Vectors pin behaviour; they are not a security proof
qodiqa-verify	Artifacts, evidence bundles, audit chains, proofs, receipts, anchors, agent runs	Node built-ins and the Python standard library (one script needs cryptography or PyNaCl); a test checks that nothing outside the repository is imported	VALID means consistent and signed by the keys you trusted
TLA+ models	Request lifecycle, nonce across a Redis restart, delegation, action chain	TLC model checker; each model has a negative control that must fail	Small bounds; not a proof that the code matches
Generated contracts	HTTP surface and SDK tables	OpenAPI generated from routes and diff-checked in CI	Some 429 and 405 responses are under-documented
qodiqa-anchors	Append-only anchor statements	Branch protection; positions never go backwards; npm test	Witness independence is a deployment fact

VALID means

Every entry reproduces its canonical form and hash, each prevHash links back to GENESIS, and, when supplied, anchors match and transparency entries are hashed, linked and signed.

It does not mean

That unhashed fields are genuine, or that a writer-capable party did not regenerate the whole chain: the audit hash is unkeyed, so that needs an anchor, an independent tail or signed leaves.

Verdicts

Exit 0 is VALID and 1 is INVALID. A proof verified without an anchors repository exits 3, VALID_PARTIAL: it checks out, but nothing outside the gateway vouches for the checkpoint.

```
node bin/verify.bundle.external.js --keys examples/front-half/keys.json \
--transparency-keys examples/front-half/transparency-keys.json examples/front-half/bundle-with-proof.json
```

Run from a clone of qodiqa-verify. The keys in examples are throwaway trust anchors; in a real audit obtain keys out of band (the gateway's /.well-known/qodiqa/keys over a channel you trust), never from the bundle being checked.

3.4 CAPABILITIES BY STATUS

Implemented, specified, planned, out of *scope.*

AREA	IMPLEMENTED	SPECIFIED	PLANNED	OUT OF SCOPE
Artifacts	Ed25519, 60 s, single use, bound to method, path and body; key lifecycle; per-artifact revocation	QODIQA-CORE-2026-001, QODIQA-SEC-2026-001	None listed	Header and query binding under V1
Policy	Eleven rule types, ten of which decide a request; consent states; four-eyes publication; simulation	Fail-closed policy evaluation; 68-point framework (FRAMEWORK-2026-003)	Retention policy type; restart-safe revocation de-duplication	Legal assessment; GDPR work is technical controls only
Screening	13 rule classifiers, four tiers; embedding layer whose resources are not distributed	Excluded from the Core decision (CORE 1.6)	None listed	A protection guarantee; reliable paraphrase or non-English detection
Execution	Guarded provider path; routing, budgets, one failover hop; streaming with check-then-release	QODIQA-ARCH-2026-001, QODIQA-RIS-2026-001	None listed	Real-service validation of hosted adapters; traffic that bypasses the gateway
Evidence	Audit chain v7 (v8 optional); RFC 6962 log; receipts; proofs; signed checkpoints	QODIQA-AUDIT-2026-001, QODIQA-ARP-2026-001	None listed	Proof a decision was right; durability across sudden power loss
Anchoring	Witnesses, append-only export, anchors repository; required at production boot	Not mapped here	None listed	Distributed mode has no checkpoints or anchors yet
Agents, MCP	Sessions, action artifacts, delegation up to 8 links, approvals, HTTPS MCP gateway, kill switch	Protocol V2 and delegation specs in the gateway repository	None listed	stdio MCP; streaming results; signed tool receipts
Identity	Signing keys, RBAC, TOTP; OIDC, SAML, LDAP against test infrastructure; per-tenant keys	QODIQA-SBTM-2026-001	None listed	Row-level security as a boundary; production identity providers
Operations	One instance with PostgreSQL and Redis; Kustomize, Helm, Terraform, Compose, systemd; backups; metrics	QODIQA-INTEROP-CONS-2026-001	Persistent anomaly baselines	Clustering, failover, throughput figures, hosted sandbox
Verification	Offline verifiers in JavaScript and Python; protocol vectors; four TLA+ models	CTSS, CVS and CERT define conformance testing, verification and certification	None listed	Any certification or attestation; proof that code matches the models

Implemented rows come from the gateway and verifier documentation; Specified from the corpus; Planned from the roadmap; Out of scope from the scope and limitations page. Corpus identifiers abbreviated: QODIQA-<name>-2026-001.

3.5 GLOSSARY

Terms used in this *document.*

ARTIFACT
A signed document that authorizes one request: five fields, Ed25519, valid for at most 60 s.

REQUEST BINDING
The hash of method, path and body carried inside the signed payload and recomputed on arrival.

NONCE
The artifact's requestId, registered once in Redis; a second use is a replay.

TENANT
A customer or environment isolated inside the gateway, with its own keys, policy and consent.

CONSENT
The record that the data subject allows a use; checked on every request.

POLICY SET
A tenant's rules, evaluated in priority order and hashed into policyHash.

FAIL CLOSED
On doubt or error, refuse: a failure produces a BLOCK, never a silent pass.

CANONICALIZATION
One byte-exact serialization of JSON used for every hash and signature.

DOMAIN SEPARATION
A distinct tag prefixed to each hash class so signatures do not cross classes.

KEK
Key-encryption key that wraps the Ed25519 private keys at rest.

AUDIT CHAIN
Hash-linked decision log: each entry includes the hash of the previous one.

INTENT ENTRY
A chain entry written before the provider call, so a crash still leaves a trace.

TRANSPARENCY LOG
A Merkle tree over entry hashes with signed roots and inclusion proofs.

CHECKPOINT
A signed snapshot of the chain head, length and previous checkpoint.

WITNESS
A separate process with its own key and log that cosigns checkpoints.

ANCHOR
A checkpoint statement stored outside the gateway, append-only.

RECEIPT
A signed statement that the gateway recorded a decision.

PROOF
One request's artifact, entries, checkpoint, inclusion proof and receipt in one document.

FOUR-EYES
Policy publication that needs draft, review and approval by different operators.

DELEGATION
A signed grant of part of an agent's authority; down a chain it can only shrink.

MCP
Model Context Protocol: the interface to tool servers, reached here only through the tool gateway.

DLP
Data loss prevention: pattern and checksum detection of personal data and secrets.

3.6 VERIFY THIS DOCUMENT

Check the standard against its published *hash.*

This overview points to the QODIQA Standard, document **QODIQA-STD-1.0**, distributed as the file QODIQA-STD-1.0.pdf. Its SHA-256 hash is **published in the release notes and shown below**.

Compute the hash of your copy and compare it with the release notes. Take the expected value from the release notes, not from the place you downloaded the file.

A match shows the file is the published file. It says nothing about whether its content is right.

EXPECTED SHA-256 OF QODIQA-STD-1.0.PDF

`eacc2276494f294921b99b7b195988657c0ce9a95c84f6f0ccfeb2287a7024ad`

WINDOWS · COMMAND PROMPT

```
certutil -hashfile QODIQA-STD-1.0.pdf SHA256
```

WINDOWS · POWERSHELL

```
Get-FileHash QODIQA-STD-1.0.pdf -Algorithm SHA256
```

MACOS AND LINUX

```
shasum -a 256 QODIQA-STD-1.0.pdf
```

Compare the 64 hexadecimal characters, ignoring case and spaces.

GATEWAY	VERIFIERS	ANCHORS	SPECIFICATION
d4bdebb91710	fe774273cc55	4d36d4e37ae1	d3b5066a29b1

CONFIRM A REVISION IN YOUR CLONE

```
git rev-parse HEAD
```

Revisions: the public repository commits this overview was written against. The command prints the full commit; its first characters should match.

QODIQA

Verify first. Authorize first. Execute second.

WWW.QODIQA.COM · CONTACT@QODIQA.COM