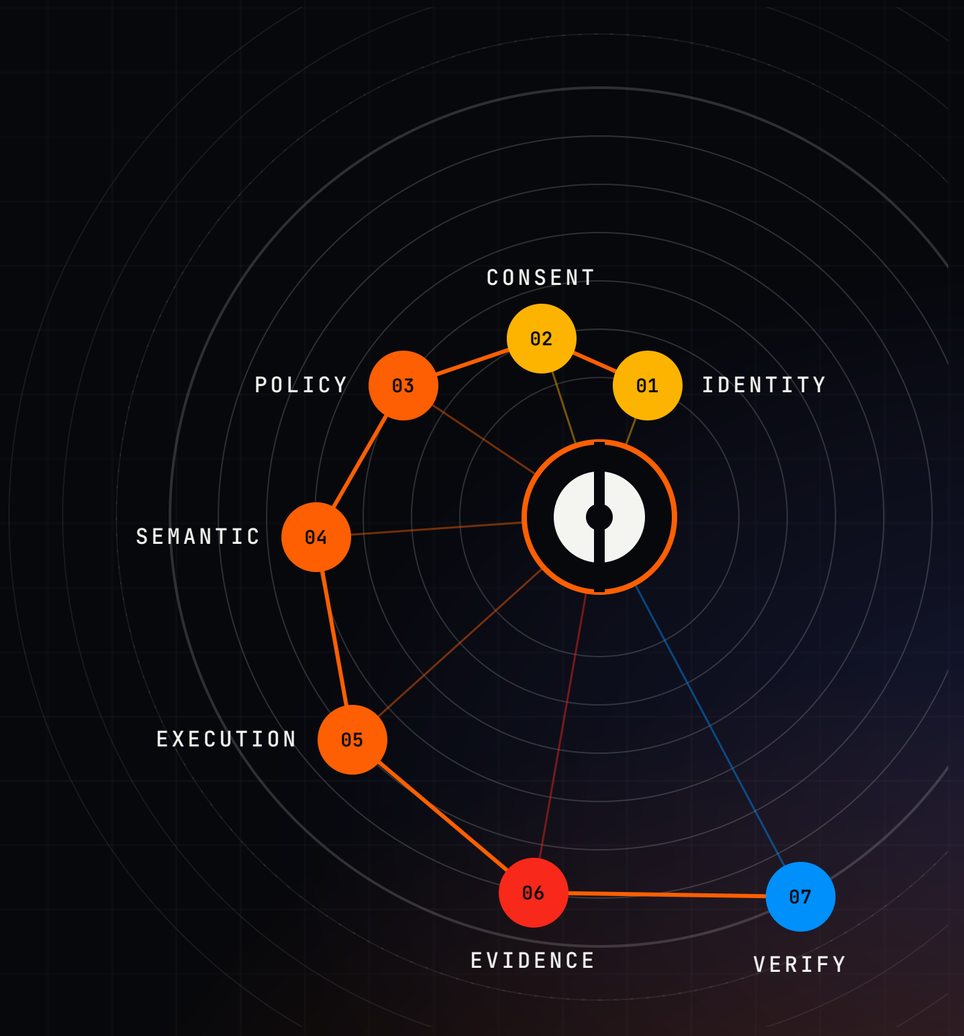
AI ENFORCEMENT AND CRYPTOGRAPHIC EVIDENCE INFRASTRUCTURE

The enforcement layer between applications and AI *execution*

Every request is verified, judged against consent and policy, inspected, and recorded as signed, independently verifiable evidence.



PRODUCT CAPABILITIES | OCTOBER 2026

02 CONTENTS

The complete product, in five *parts.*

What QODIQA is, how it decides, how it proves, how it is run, and how the capabilities fit together.

01 The product

Contents 2
The central idea 3
The platform at a glance 4

02 How the system works.

System architecture 6
The execution lifecycle 7
Consent and policy enforcement 8
Policy rule types and decision order 9
Semantic intelligence and risk classification 10
Data-loss prevention and content controls 11
Execution, routing and AI providers 12
Streaming and response control 13
Replay protection, anomaly detection and idempotency 14
Policy simulation, governance and provenance 15
API conventions: versioning, pagination and errors 16
Limits, quotas and budgets 17
Platform runtime and hardening 18

03 Trust and security.

Cryptographic trust architecture 20
Key lifecycle, rotation and custody 21
Identity, access and tenancy 22
Single sign-on, MFA and operator sessions 23
Operator access and administration 24
Evidence, provenance and verification 25
The audit chain in depth 26
Transparency log, receipts and proofs 27
Independent and offline verification 28
Protocol V2, canonicalisation and the MAC keyring 29
Transparency gap detection, witnesses and external anchors 30
Retention, legal holds and data-subject rights 31
Agents, tools, actions and MCP 32
Action authorisation and agent plans 33

04 Enterprise platform.

APIs and developer integration 35
SDKs and client behaviour 36
Tenant portal and operator consoles 37
Attestation, certification and compliance reporting 38
Webhooks and usage metering 39
Data architecture and persistence 40
Backup, restore and disaster recovery 41
Observability and operational intelligence 42
Alerting, incidents and escalation 43
Deployment and integration architecture 44

05 The complete platform.

Capability map 46
An illustrative execution scenario 47
Integration and interoperability model 48
Why the architecture matters 49
In summary 50

03 THE CENTRAL IDEA

From requesting a response to governing an *execution*.

Most applications call a model and trust the controls around it. QODIQA makes those controls part of the execution path itself.



The QODIQA execution path. A blocked request never reaches the model, and every outcome leaves a record.

01 · DECIDE FIRST

Controls run before the model does.

Identity, consent, tenant policy and content inspection are evaluated deterministically before any provider call. The default answer to uncertainty is **block**.

02 · BIND THE DECISION

Authorisation travels with the request.

A short-lived Ed25519 artifact binds the exact request to the tenant's policy set. A request that differs from what was authorised does not execute.

03 · LEAVE EVIDENCE

Every outcome can be checked later.

Decisions are written to a hash-chained audit log and a Merkle transparency log, so a third party can verify what was decided without trusting the operator.

04 THE PLATFORM AT A GLANCE

Twelve capabilities. One execution *path.*

Enforcement decides, trust makes the decision provable, and operations keep it running.

ENFORCE — Decide before execution

01
Gateway and API
A hardened edge, a generated contract and an ordered request lifecycle.

02
Consent and policy
Per-actor consent with revocation, and eleven policy rule types in priority order (ten decide a request; DATA_RETENTION declares retention settings).

03
Semantic inspection
Rule-based classifiers and an embedding ensemble judge prompts, responses and tool results.

04
Execution and routing
One guarded path to eight provider families, with breakers, retries and failover.

TRUST — Make decisions provable

05
Identity and tenancy
Operators, machine identities, organisations and tenants with RBAC, SSO and MFA.

06
Cryptographic artifacts
Ed25519 signing, domain separation and optional KMS or HSM key custody.

07
Evidence and transparency
Hash-chained audit records, signed checkpoints and a Merkle log with proofs.

08
Agents, tools and MCP
Authorised, signed and recorded actions with approvals and delegations.

OPERATE — Keep it running

09
State and persistence
PostgreSQL with row-level tenant isolation, Redis for live state, file-based stores.

10
Observability
Structured logs, Prometheus metrics, tracing, health probes and seven alert channels.

11
SDKs and tooling
JavaScript, Python, Go and Java clients, offline verifiers and a local sandbox.

12
Deployment
Hardened images, Compose, Kubernetes, Helm, Terraform and backup and recovery tooling.

PART 02

02

How the system *works.*

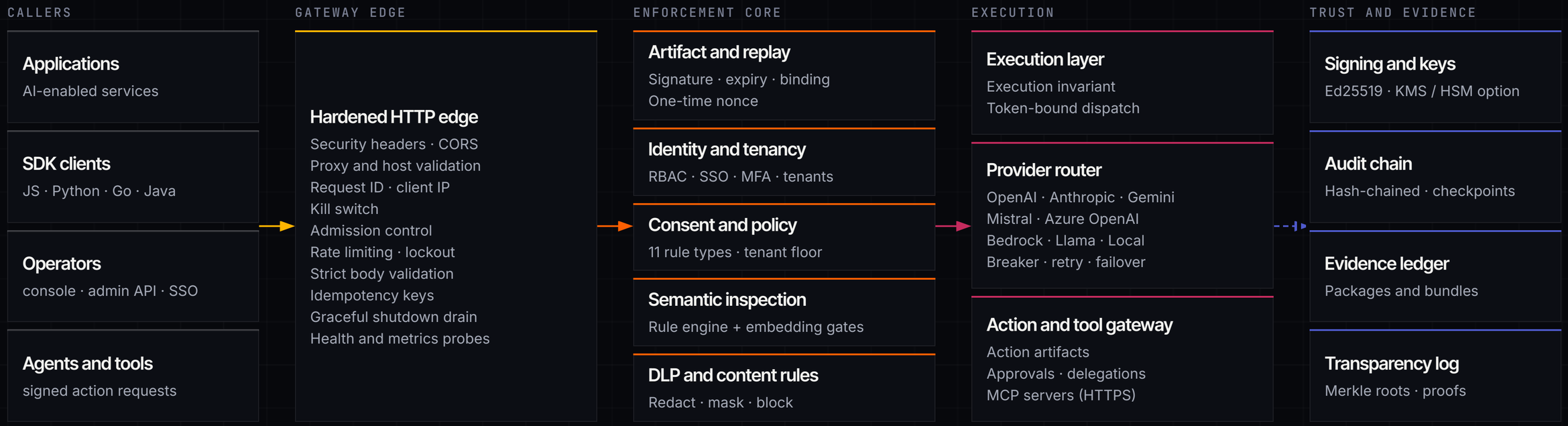
The architecture, the request lifecycle, and the controls that decide before anything executes.

06 System architecture
07 The execution lifecycle
08 Consent and policy enforcement
09 Policy rule types and decision order
10 Semantic intelligence and risk classification
11 Data-loss prevention and content controls
12 Execution, routing and AI providers
13 Streaming and response control
14 Replay protection, anomaly detection and idempotency
15 Policy simulation, governance and provenance
16 API conventions: versioning, pagination and errors
17 Limits, quotas and budgets
18 Platform runtime and hardening

06 SYSTEM ARCHITECTURE

Five layers between a caller and a *model.*

Requests enter a hardened edge, pass an enforcement core, reach one guarded execution layer, and emit evidence on the way out.



FOUNDATION

State
PostgreSQL · row-level tenant isolation
Redis · replay, limits, revocation
File stores · keys, policies, audit log

Key custody
Local · Vault · AWS · Azure
Google Cloud · PKCS#11 HSM

Observability
Logs · metrics · traces
Health · alerts · dashboards

Offline verification
Standalone verifiers · SDK helpers
Published public key sets

07 THE EXECUTION LIFECYCLE

Twelve ordered stages from request to *response.*

The machine path for an AI chat request. Each stage can end the request; none can be skipped.

01
Gateway edge
Kill switch, rate limit and admission control

02 BLOCK
Request validation
Strict schema, size and content-type checks

03 BLOCK
Artifact verification
Signature, expiry, request binding, revocation

04 BLOCK
Replay and anomaly
One-time nonce and burst detection in one atomic step

05 BLOCK
Tenant and identity
Tenant, organisation and signing identity still active

06 BLOCK
Consent and policy
Consent record and rules in priority order

07 BLOCK
Semantic and DLP
Rule engine, embedding gates, content rules, DLP

08 BLOCK
Intent evidence
Intent written to audit chain and transparency log first

09 BLOCK
Provider call
Execution token, routing, retry and failover

10 BLOCK
Output inspection
Size cap, semantic check and DLP on the response

11 BLOCK
Decision evidence
Audit entry, transparency leaf and signed receipt

12
Response
EXECUTE with artifact, key and proof references

Deny, never guess

Every enforcement refusal is a BLOCK decision with a reason code from a published catalogue. Missing state or a failed dependency blocks.

Not every request is identical

Streaming requests release checked output in batches. The `/verify` dry run applies the same checks with no provider call and no evidence.

Failures stay distinct

Throttles return 429, overload and unavailable dependencies 503, provider timeouts 504. Retry hints accompany transport refusals.

08 CONSENT AND POLICY ENFORCEMENT

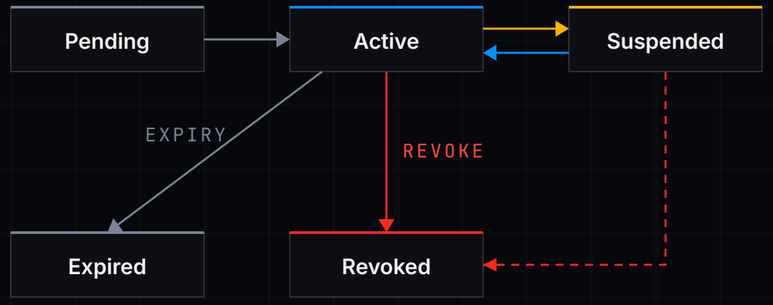
Consent is who agreed. Policy is what is *allowed.*

Two distinct controls, both evaluated before a request can execute. A tenant can tighten them but never loosen them.

CONSENT VALIDATION

A record ties a caller, an AI provider and a tenant to a scope and an expiry. Every request is checked in a fixed order.

- **Revocation sync** must be healthy, or the request is denied
- **Sealed cache** lookup, then database lookup
- **State** must be Active, and **scope** must match
- **Provenance snapshot** recorded without personal data
- **Operations**: grant, revoke, suspend, resume, revoke all for an actor



POLICY EVALUATION

Policies are validated, frozen and evaluated in priority order. The first failing rule blocks and stops evaluation.

Eleven rule types, from consent and network rules to attribute, residency, content, budget and routing rules, are set out under Policy rule types.

- **Tenant floor**: global rules cannot be disabled or loosened
- **Missing evaluator, error or empty set**: all block
- **Maker-checker** governance: draft, review, approve, publish by different people
- **Emergency override**: incident-referenced, time-limited, retro-approved

OUTCOME

EXECUTE or BLOCK.

The decision records the policy version, the deciding rule and the consent provenance.

ALWAYS ENFORCED

A policy set without an enabled consent rule is refused at load, and consent checks are never served from a decision cache.

OPERATOR CONTROL

Consent management requires an admin permission plus second-factor step-up.

09 POLICY RULE TYPES AND DECISION ORDER

Eleven rule types. One deterministic *order.*

Rules are validated, frozen and sorted by priority. The first failure blocks, and nothing is left to chance.

	RULE TYPE	WHAT IT ENFORCES	EVALUATED IN
01	REQUIRE_CONSENT	The caller must hold an active consent for this provider and scope	Policy engine
02	BLOCK_IP	Exact IPv4 addresses, IPv4 ranges and IPv6 ranges are refused	Policy engine
03	TIME_WINDOW	Execution is allowed only inside configured UTC days and times	Policy engine
04	RATE_LIMIT	Per-user or per-IP counters; closed if the counter store fails	Policy engine
05	ABAC	Attribute conditions that deny or require, over subject, tenant, provider, time, network, tool and agent attributes; applies to requests and tool calls	Policy engine
06	DATA_RESIDENCY	The chosen provider must map to an allowed region	Policy engine
07	DLP	Direction (input, output, tool input, tool output), rules and minimum confidence	Dedicated stage
08	CONTENT_RULE	Tenant-defined rules over input and output content	Dedicated stage
09	BUDGET	Limits on requests, tokens and cost	Dedicated stage
10	ROUTING	Candidate providers and models, with a selection strategy	Dedicated stage
11	DATA_RETENTION	Declares retention settings; validated when policies load	Declaration only

DECISION ORDER

- Priority from 0 to 999,999; lowest evaluated first
- The first failing rule blocks and stops evaluation
- Consecutive rate rules run in one batched round trip
- An empty or fully disabled policy set blocks
- A missing evaluator or a thrown error blocks
- Eleven rule types, ten of which decide a request; DATA_RETENTION declares retention settings

CONTEXT CHECKS FIRST

Before any rule runs, the request context is validated: wrong types, oversize values, prototype-pollution keys and implausible timestamps all block.

OUTCOME AND RECORD

EXECUTE or BLOCK, with the policy version, the deciding rule and every rule evaluated. An optional decision cache covers only pure rules and never consent.

10 SEMANTIC INTELLIGENCE AND RISK CLASSIFICATION

Two engines judge *meaning* before anything executes.

Rule-based classifiers and an embedding ensemble inspect prompts, responses and tool results, and share one verdict shape.

Normalise 01
Decode base64, hex, URL and HTML escapes; fold look-alike letters; strip invisible characters

Rule engine 02
13 classifiers score intent; confidence times a severity weight; the highest score decides

Embedding ensemble 03
In-process ONNX encoder, 400-character windows; five gates combined with OR

Intent gate 04
Recognises protective or defensive framing, then re-scores decoded readings of the text

Verdict 05
ALLOW or DENY with a reason and evidence, handed to the enforcement pipeline

THE FIVE EMBEDDING GATES — ANY ONE FIRING BLOCKS

Anchors
Similarity to curated known-bad examples across six families

Head
A trained classifier over the embedding

v9
A classifier over the embedding plus 21 surface features

v5
A second embedding-only classifier

Fine-tuned encoder
Consulted when anchor similarity enters a low band, which bounds latency

RULE-ENGINE CLASSIFIERS

CREDENTIAL EXTRACTION · DATA EXFILTRATION · FILE-SYSTEM ENUMERATION · IMPERSONATION · JAILBREAK · MEMORY EXTRACTION · PII EXTRACTION · POLICY BYPASS · PROMPT INJECTION · SECURITY OVERRIDE · SYSTEM-PROMPT EXTRACTION · TOOL ABUSE · UNAUTHORIZED TOOL REQUEST

Semantic assessment is a risk signal
Scores and thresholds are calibrated offline, run with no model call or network access, and fail closed: an engine failure becomes a block. Tenants can only tighten thresholds. A score is a judgement about meaning, not a proof.

Deterministic enforcement is the guarantee layer
Signature, binding, replay, consent and policy checks are exact and repeatable. The semantic verdict feeds this pipeline but never replaces it: an embedding allow cannot override a rule-engine block or a failed signature.

11 DATA-LOSS PREVENTION AND CONTENT CONTROLS

Sensitive data is found, redacted or *stopped.*

An optional enterprise module scans prompts, responses and tool traffic for personal data and secrets before they cross a boundary.

24 DETECTORS IN FIVE CATEGORIES

EMAIL · PHONE · CARD (LUHN) · IBAN · NATIONAL IDS · SEVEN COUNTRIES · CLOUD AND PLATFORM API KEYS · JWT · CODE SECRETS · PEM PRIVATE KEYS

- **Personal:** email and phone
- **Financial:** Luhn-checked cards and IBAN
- **National IDs:** Romania, Poland, Netherlands, Germany, United States, Spain and United Kingdom
- **Credentials:** keys for AWS, GitHub, Stripe, Slack, Google, Anthropic, OpenAI, npm, SendGrid and QODIQA, plus JWTs
- **Crypto:** PEM private-key blocks

FOUR ACTIONS, APPLIED BY DIRECTION

DETECT · REDACT · MASK · BLOCK

- A DLP policy sets direction: input, output, tool input or tool output
- Redaction uses stable keyed tokens; minimum confidence defaults to 0.6
- Text is normalised first, defeating invisible characters, Unicode tricks and look-alike letters
- Data classification runs from public to restricted, with exfiltration signals for URLs, bulk records and encoded blobs
- Scans are capped near 1 MiB; truncated input is blocked when an enforcing action is set

WHERE IT APPLIES

- Chat requests and provider responses
- Tool arguments and tool results on the agent path
- Output DLP is refused on streaming requests rather than weakened

CONTENT RULES

Tenant-defined content rules run alongside DLP on input and output. Like DLP, they are configured in policy, enforced by a dedicated pipeline stage, and refused for streaming output.

12 EXECUTION, ROUTING AND AI PROVIDERS

One guarded path to every *model.*

Application code never calls a provider directly. Adapters are reachable only through the router, and only with a verified context.

Verified request 01
Passed identity, consent, policy and inspection

→

Execution invariant 02
Dispatch requires a verified context and a fresh HMAC execution token, or it is refused

→

Policy-driven routing 03
Optional: cheapest, fastest, least-loaded or declared order; any change is re-authorised

→

Provider router 04
The only route to an adapter, with a breaker and health score per provider

→

AI model 05
Hosted or local

PROVIDER ADAPTERS BEHIND ONE NORMALISED INTERFACE

OpenAI
Official SDK · streaming

Anthropic
Messages API · streaming

Google Gemini
generateContent

Mistral
OpenAI-compatible

Azure OpenAI
API key or Entra ID

Amazon Bedrock
Converse API

Llama
Operator-supplied endpoint

Local models
Any OpenAI-compatible server

Timeouts
Each call has a deadline covering headers and body: 30 seconds by default, set per provider.

Careful retries
One retry at most, only for rate limits or when no response arrived. Completions are never blindly re-sent.

Circuit breakers
A provider opens after five consecutive counted failures, then half-opens to probe recovery.

Authorised failover
Opt-in, one hop, to a configured target that is re-checked against consent, scope and policy.

Network safety and extensibility
Hosted providers are restricted to allow-listed hosts, connections refuse metadata and link-local ranges, and redirects are not followed. Administrators can add OpenAI-compatible providers at runtime; each is integrity-protected and bound to specific hosts.

13 STREAMING AND RESPONSE CONTROL

Check first. Release *second.*

Streaming responses are gated on the way out, and evidence about what was released is written when the stream ends.

Abandon check 01
Skip the provider call if the request was already answered

→

Open evidence 02
Intent and stream-opened entries are written first

→

Open stream 03
SSE headers, disclosure header, stream_started event

→

Provider deltas 04
Fresh execution token, then output arrives in pieces

→

Output gate 05
Text is held until checked; checked text is released in batches

→

Final check 06
The full text is checked once more at the end of the stream

→

Decision evidence 07
ALLOW only if completed; hash of exactly the released text

On a block

Provider connection aborted, held text discarded, and a `stream_cancelled` event carries the reason.

Non-streaming requests

The complete response is generated, then checked as a whole: size cap, semantic inspection, content rules and data-loss prevention. A blocked response is withheld entirely.

The audit entry, transparency leaf and receipt are written before the response is returned.

How a stream ends

Every stream finishes with one terminal event: `stream_completed` `stream_cancelled`, with a reason such as client disconnect, stream timeout, server shutdown, semantic block, output too large or provider error Both are followed by `[DONE]`.

What the design does and does not promise

Text is checked before it is released, in batches of about 400 characters, not token by token.

Text already released to a client cannot be recalled: the stream protocol has no retract. The final check then serves the evidence record.

Gemini and Bedrock responses arrive as one chunk.

14 REPLAY PROTECTION, ANOMALY DETECTION AND IDEMPOTENCY

Every artifact works exactly *once.*

A signed request is single-use. Bursts are detected in the same atomic step, and retries are made safe with idempotency keys.

Signed artifact 01
Unique request id (UUID v4)

→

Atomic registration 02
Set-if-absent in Redis, lifetime equals remaining validity

→

Anomaly window 03
Same step: sliding 60-second counters per key and global

→

Decision 04
First use proceeds; everything else is blocked

REPLAY DETECTED
Second use of the same request id

REPLAY STATE UNPROVABLE
Redis restarted after the artifact was issued

NONCE STORE ERROR
Store failure: fail closed, no memory fallback

ANOMALY DETECTED
More than 100 per key or 5,000 global per minute

TIMING RULES

- At most 5 seconds of future clock skew
- Maximum age of 60 seconds
- Expiry has no tolerance
- The dry-run `/verify` never consumes a nonce

SAFE BY CONSTRUCTION

- Redis persistence is checked at start; unsafe setups are refused unless strict replay is explicitly disabled
- Limits can be overridden per organisation
- Any store error blocks the request

IDEMPOTENCY KEYS

- Bound to tenant, key, caller, provider and request hash
- Outcomes: proceed, replay a stored result, conflict, in progress, or store unavailable
- Only successful executions are kept, for 24 hours by default

15 POLICY SIMULATION, GOVERNANCE AND PROVENANCE

Preview a policy change. Govern it. Trace *it.*

Policy changes are simulated without side effects, then move through a maker-checker workflow with independent approval and rollback.

01
Current or draft policies
The live policy set, or a proposed set

02
Request and context
A representative request to evaluate

03
Policy simulation engine
Same evaluators, no side effects
Every response states execution was prevented
Nothing is stored, signed or executed

04
Decision preview
EXECUTE or BLOCK with the deciding rule, for the live set or a comparison of two sets

SIMULATE
one evaluation

COMPARE
current against proposed

BATCH SIMULATE
up to 100 items per call

GOVERNED CHANGE · EACH TRANSITION IS RECORDED IN A HASH-CHAINED EVIDENCE TRAIL

DRAFT → SUBMITTED → IN REVIEW → REVIEWED → APPROVED → SCHEDULED → PUBLISHED

Also: REJECTED · WITHDRAWN · EXPIRED. Scheduling is optional. Publish re-hashes the stored payload and refuses on a mismatch.

VISUAL POLICY TOOLING
Read-only endpoints draw policies as a graph, tree or decision flow, show dependencies and statistics, highlight risk, and export. The editor writes governed change requests, never live policy.

INDEPENDENT ROLES
Reviewer, approver and publisher must differ from the author. Strict mode also separates approver from reviewer, and a lineage check blocks approvers provisioned by the author.

ROLLBACK AND EMERGENCY OVERRIDE
Rollback is requested and then approved by someone other than the requester and original publisher. An emergency override needs an incident reference, lasts 24 hours by default and must be retro-approved.

PROVENANCE
A provenance API reconstructs policy history, diffs versions, shows lineage and integrity, and reports across tenants.

16 API CONVENTIONS: VERSIONING, PAGINATION AND ERRORS

One contract. Predictable *behaviour.*

The HTTP surface is versioned, paginated and described by a generated contract, with a published reason for every refusal.

VERSIONING

- A versioned surface under `/v1`, with a version header on responses
- Aliases keep older paths working; a signed request binds the path actually used
- Deprecated routes advertise `Deprecation`, `Sunset` and successor `Link` headers

PAGINATION

- `page` from 1, `pageSize` with a default of 50 and a maximum of 500, and `sortOrder`
- Used across administrative list endpoints
- Some routes also accept `limit` and `offset`
- Unbounded results are capped

DECISIONS AND ERRORS

- Each reason code carries an HTTP status, a category and a retryable flag
- Categories span input, validation, replay, policy, consent, provider, budget, semantic and authentication
- The catalogue is published in the OpenAPI document

RESPONSE SHAPES

- Machine surfaces return a decision body: BLOCK with a reason code and request id
- Administrative routes return an envelope with status, timestamp and error
- A block maps to HTTP 200 by default, or to 4xx and 5xx by configuration

STRICT VALIDATION

- Unknown fields, oversize input and malformed canonical forms are rejected
- Reason codes travel from the schema to the response
- Request bodies, headers and artifacts have size limits

COMPATIBILITY POLICY

- Error codes never change so that a valid V1 artifact becomes invalid
- A tenant can require a minimum protocol version
- Contract, SDK tables and protocol vectors are generated from code and checked for drift

17 LIMITS, QUOTAS AND BUDGETS

Nine layers of protection before the model *runs*.

Rate, capacity, cost and classification limits keep every tenant inside defined bounds, and each one refuses rather than degrades silently.

	CONTROL	SCOPE	BEHAVIOUR
01	**Rate limiting**	Per surface, with tenant, user, key and endpoint policies	Sliding window with burst; HTTP 429 with RateLimit headers; closed by default when Redis is down
02	**Admission control**	Sign, chat, verify, actions and tools	In-flight limit plus a bounded queue; HTTP 503 with Retry-After before heavy work
03	**Organisation quotas**	Per organisation, applied to its tenants	Per-minute and per-key request limits set the organisation detection thresholds; streaming-session and retention settings are recorded on the organisation
04	**Anomaly detection**	Per signing key and global	100 requests per key and 5,000 globally per minute by default; blocked as a burst
05	**Classification budget**	Per tenant	Token bucket of classification windows with refill and burst limits; HTTP 429 when exhausted
06	**Budget policy rule**	Per tenant policy	Requests, tokens and cost, reserved before the provider call and settled after; HTTP 429 per dimension
07	**Output size cap**	Provider output	Responses above 65,536 characters by default are blocked as too large
08	**Authentication lockout**	Client IP and credential	Five failures lock for 15 minutes with exponential backoff; closed if the store is down
09	**Kill switch**	Platform-wide	Refuses signing, chat, actions, delegations and tools with HTTP 503; closed if state is unreadable

Usage metering is read-side aggregation of token usage per tenant and UTC day. There is no write-time metering or billing in the product.

18 PLATFORM RUNTIME AND HARDENING

A gateway that refuses to start *unsafe.*

Production boot is validated against declarative rules, the edge is hardened, and shutdown drains requests and signs a final checkpoint.

Process start 01
Hardening, configuration validation, key store checks, policy load

Before listening 02
Evidence durability, control plane, mirror recovery, checkpoint check, classifier readiness, Redis persistence, revocation subscribers

Serving 03
Ready only when all three Redis stores are ready; a secret-free boot report is emitted

Graceful shutdown 04
Readiness drops, new requests get 503, streams cancelled, requests drain, final signed checkpoint, stores close

BOOT VALIDATION

- Fatal violations are listed together before exit; warnings are logged
- Production needs distinct 64-hex secrets, TLS to PostgreSQL and Redis, key files outside the application, a policy file and MFA on
- No migration credential may be present in the gateway

EDGE HARDENING

- Security headers, strict CSP and HSTS in production
- Proxy chain depth, forwarded-address and host validation
- Fail-closed CORS allow-list; request id on every response
- Body limit of 64 KiB by default; strict UTF-8

PROCESS SAFETY

- Restrictive file creation mask and crash handlers
- Refuses to run as a cluster worker
- Single-writer audit chain: one active process per deployment
- Deployment mode single-node or distributed, validated at start
- Background tasks run under a registry with timeout and retry

SECRETS AND KEYS

- Key and config-secret providers initialise before the server
- Config secrets from the environment or HashiCorp Vault KV
- Zeroisation of the key-encryption key at shutdown

PART 03

03

Trust and *security*.

Cryptographic authorisation, identity, verifiable evidence and governed agent actions.

20 Cryptographic trust architecture
21 Key lifecycle, rotation and custody
22 Identity, access and tenancy
23 Single sign-on, MFA and operator sessions
24 Operator access and administration
25 Evidence, provenance and verification
26 The audit chain in depth
27 Transparency log, receipts and proofs
28 Independent and offline verification
29 Protocol V2, canonicalisation and the MAC keyring
30 Transparency gap detection, witnesses and external anchors
31 Retention, legal holds and data-subject rights
32 Agents, tools, actions and MCP
33 Action authorisation and agent plans

20 CRYPTOGRAPHIC TRUST ARCHITECTURE

Signed authorisation, bound to the exact *request.*

A short-lived Ed25519 artifact ties a request to the tenant's policy set. The execution layer will not run without it.

Canonical request 01
Method, path and body in a stable, strict JSON form

→

Domain-separated hash 02
SHA-256 over a class tag and the canonical bytes, so signatures cannot cross object types

→

Ed25519 signature 03
Signed with the tenant's artifact key, locally or inside Vault Transit or a PKCS#11 HSM

→

Signed artifact 04
Payload, key id, timestamp, expiry and signature, valid for at most 60 seconds

→

Verification 05
Schema, signature, key lifecycle and time checked; the result is EXECUTE or BLOCK

THREE INDEPENDENT TRUST DOMAINS

Artifact keys
Per tenant. Sign request artifacts, action artifacts, approvals and delegations.

Transparency keys
Sign transparency entries, Merkle roots and receipts.

Checkpoint keys
Sign audit checkpoints and witness co-signatures.

Key custody and lifecycle
Private keys encrypted at rest (AES-256-GCM) and sealed. Optional Vault, AWS KMS, Azure, Google Cloud or HSM custody. Graceful and emergency rotation; deprecated keys verify only older artifacts.

Cryptographic integrity is not a statement about truth

A valid artifact proves that this request, under this policy, was authorised by a holder of this key at this time, and that the record has not been altered since.

It does not prove that an AI response is factually correct, safe or compliant.

QODIQA keeps authorisation, enforcement and evidence separate, so each claim stays precise.

21 KEY LIFECYCLE, ROTATION AND CUSTODY

Keys that rotate, retire and can be *revoked.*

Three independent trust domains each have their own keys, rotation schedule and optional external custody.



LOCAL (DEFAULT)	Encrypted key files, software signing
HASHICORP VAULT KV	Supplies the key-encryption key
VAULT TRANSIT	Native Ed25519 signing; key never leaves Vault
AWS KMS	Wraps the key-encryption key
AZURE KEY VAULT	Wraps the key-encryption key
GOOGLE CLOUD KMS	Wraps the key-encryption key
PKCS#11 HSM	Native key generation and signing

ROTATION: GRACEFUL OR EMERGENCY · 90-DAY DEADLINE · 180-DAY EXPIRY · EXPIRY WARNINGS

AUTOMATIC ROTATION

- On by default, monthly unless set daily, weekly or disabled
- A Redis lock with a fencing token; closed if Redis is unavailable
- Followers reload keys every five minutes
- A forward-only key-status timeline retires a key on every node

KEY ADMINISTRATION

- Inventory, health and status; graceful and emergency rotation; revoke and recover
- Key-encryption-key and MAC keyring rotation; purge of expired keys
- Permissioned and stepped up; concurrent rotations return 409

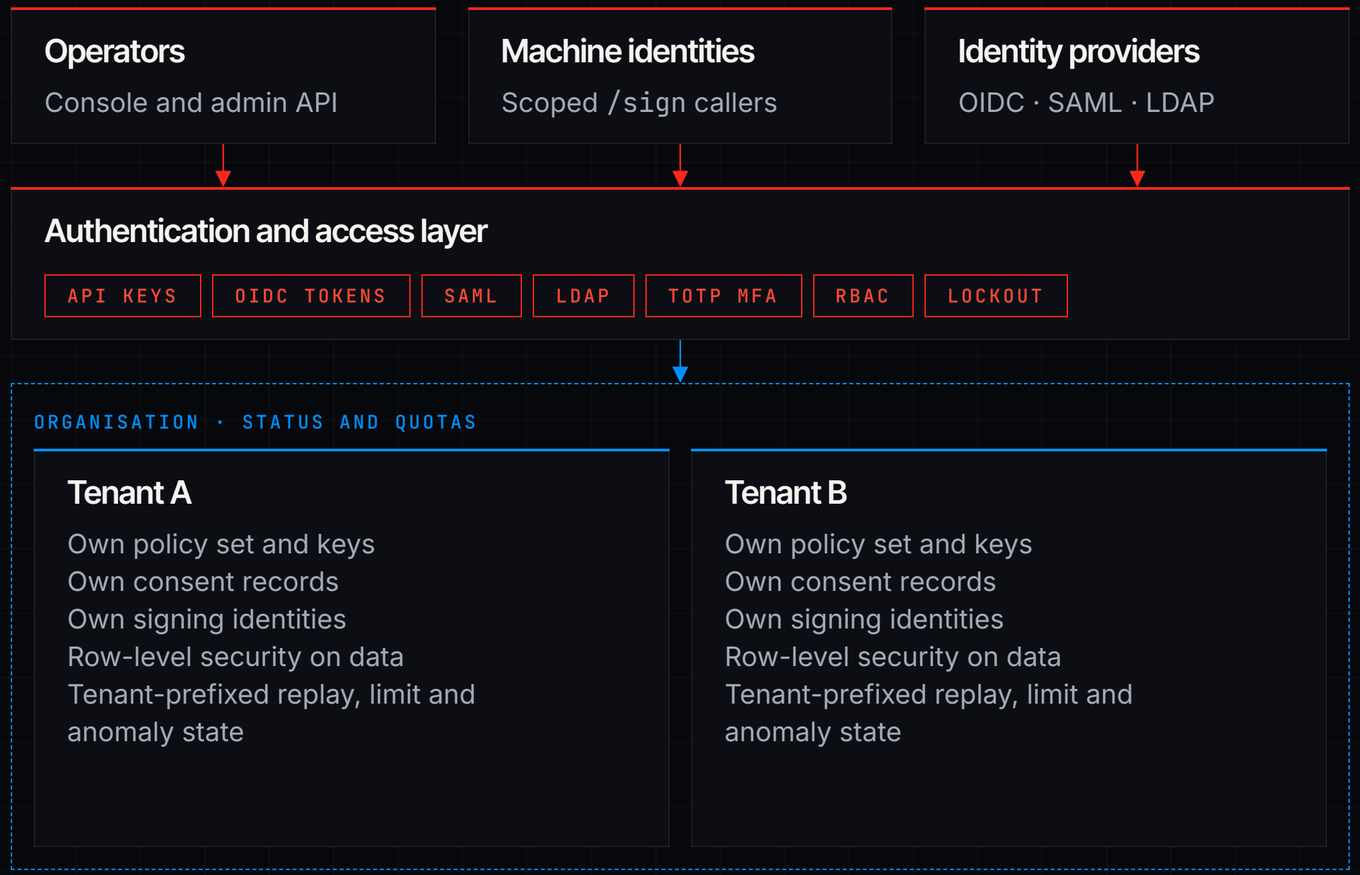
PROTECTION AT REST

- Private keys encrypted with AES-256-GCM and sealed against tampering; a plaintext key aborts start
- Application field encryption (AES-256-GCM per purpose) is required in production
- Only artifact keys use native signing; evidence seals are omitted under native providers

22 IDENTITY, ACCESS AND TENANCY

Every request is attributed, scoped and *isolated.*

Identity decides who is calling. Tenancy decides whose data, policies and keys apply.



AUTHENTICATION

Operators authenticate by federated token, per-operator API key or session; TOTP step-up protects state-changing calls. SSO is optional and activates only when configured.

AUTHORISATION

Six built-in roles over a fixed permission set, plus namespaced custom roles. Governance duties are split between proposer, reviewer, approver and publisher.

ORGANISATIONS AND QUOTAS

An organisation groups tenants and carries a status of active, suspended or disabled. Status changes reach every replica through signed events, and an organisation can set its own request thresholds.

23 SINGLE SIGN-ON, MFA AND OPERATOR SESSIONS

Enterprise identity, without *shortcuts.*

OIDC, SAML and LDAP are optional. Each activates only when configured, and a configured provider that cannot load stops the start.

OPENID CONNECT

- Signature algorithms allow-listed (RSA, PSS, ECDSA, EdDSA); none and HMAC never accepted
- Issuer, audience, expiry, skew and maximum lifetime checked
- JWKS fetched over HTTPS with SSRF guards and stale-if-error caching
- Assurance claims honoured for MFA step-up
- Serves operators and machine identities

SAML 2.0

- Signed relay state; SHA-2 signatures only
- DOCTYPE and entity declarations rejected
- Each assertion can be used once; if the replay store fails, sign-in is refused

LDAP AND ACTIVE DIRECTORY

- LDAPS or StartTLS at TLS 1.2 or later
- Empty passwords refused; injection-safe filters
- The user proves the password by binding as themselves
- Required groups gate admission; five failures in 15 minutes lock

OPERATOR SESSIONS

- Signed tokens, 30 minutes by default (60 seconds to 8 hours)
- Re-checked against the user on every request
- Disabling or unlinking a user ends their sessions at once

MFA WITH TOTP

- RFC 6238, six digits; secrets encrypted at rest
- Ten hashed recovery codes
- Five failures lock the user for 15 minutes
- Required by default; production refuses to disable it

SCOPE OF SSO

- Users are linked to an account in advance
- No SCIM, just-in-time provisioning or group-to-role mapping
- A provider-assured session satisfies step-up

24 OPERATOR ACCESS AND ADMINISTRATION

Operators authenticate three ways. One is *deprecated*.

Administrative access is permissioned, stepped up for changes, and kept separate from machine callers and tenant portal users.

01

Federation token

OIDC or SAML, and LDAP session tokens, mapped to an enabled RBAC user

02

Per-operator API key

Hashed at rest, with expiry warnings and rotation with a grace period

03 · DEPRECATED

Shared operations token

Deprecated. No user identity, logged on every use. Replaced by 01 and 02

EVERY ADMIN CALL

LOCKOUT PRE-CHECK → IP ALLOW-LIST → AUTHENTICATION → RBAC PERMISSION → MFA STEP-UP FOR MUTATIONS

SIX BUILT-IN ROLES

ADMIN	all 66 permissions
COMPLIANCE	34 permissions
AUDITOR	28 permissions
OPERATOR	16 permissions
TENANT	4 read-only permissions
VIEWER	dashboard only

Custom roles are namespaced and built from the same fixed permission set. The last admin cannot be removed; the first admin key comes from an operator bootstrap command.

STEP-UP AND LOCKOUT

- Mutating calls need a TOTP or recovery code in a request header
- Provider-assured SSO sessions satisfy step-up
- Five failures lock for 15 minutes; backoff 250 ms to 5 s
- Production refuses to disable MFA

SEPARATE CREDENTIAL FAMILIES

- **Signing identities**: API keys or OIDC-linked, scoped to tenant, environment and provider, one permission
- **Tenant portal users**: signed session tokens
- **Agents**: gateway-issued identities with optional proof of possession

CONTROL PLANE

- Tenants, roles, policies, identities and key status replicate from a versioned PostgreSQL store
- Reads older than 5 seconds are refused
- Writes use optimistic versioning, so conflicting changes are rejected

25 EVIDENCE, PROVENANCE AND VERIFICATION

Three records. One verifiable *story.*

A decision is written to a hash-chained audit log, an evidence ledger and a Merkle transparency log, each checkable on its own.

AUDIT CHAIN

EVERY ENTRY CARRIES THE HASH OF THE ONE BEFORE IT

Decision entry
Intent, decision and commit records, written first

→

Hash link
Each entry hash covers its content and its predecessor

→

Signed checkpoint
Every 1,000 entries or 15 minutes, signed and chained

→

Witness and anchor
Optional co-signing and export to external timestamping

EVIDENCE LEDGER

PER-DECISION RECORDS OF WHAT WAS CHECKED

Evidence record
Artifact, policy, consent, caller and audit-record hashes

→

11-step check
Hashes, chain continuity and decision consistency

→

Evidence package
On demand per artifact, with an Ed25519 seal

→

Evidence bundle
Package plus inclusion proof; JSON, PDF or CSV

TRANSPARENCY LOG

A MERKLE LOG OF DECISIONS

Log leaf
Each decision registers a signed, chained entry

→

Merkle tree
RFC 6962 construction with block-level subtrees

→

Signed root
Roots generated on a schedule and chained

→

Receipt and proofs
Signed receipts, inclusion and consistency proofs

Verification

Evidence can be checked by independent verifiers that need no QODIQA runtime.

Trust comes from published key sets or keys supplied out of band, never from the bundle.

The methods are set out under Independent and offline verification.

Distinct jobs

The chain orders events, the ledger explains decisions, and the log lets outsiders check inclusion.

Distinct limits

Witnessing and anchoring are optional and off by default. Transparency roots are covered through the checkpoints that embed them.

What a proof shows

That a decision was recorded and not altered since. It is evidence about enforcement, not about whether a model answer was true.

26 THE AUDIT CHAIN IN DEPTH

A chain that is written first, verified *always.*

Decisions are recorded before the provider is called, and the chain is checked at start, continuously and at every checkpoint.

Entry built 01
Intent, decision, stream and commit records

→

Write-ahead log 02
Durable before it is acknowledged

→

Append to chain 03
Hash covers content and the previous entry

→

Checksum anchor 04
Written to a file and to PostgreSQL

→

PostgreSQL mirror 05
Serialised, back-pressured

WRITE PATH AND MODES

- Single node by default: log, chain file, anchor, then mirror
- Optional distributed mode: PostgreSQL orders entries and enforces replay uniqueness
- Intent, decision and stream-opened entries precede the provider call; a failed audit write fails the call
- Identities are pseudonymised before entering the chain

VERIFICATION

- Boot walks the whole chain and refuses to start on a violation
- Live verifier: incremental every 60 seconds, full every 4 hours, PostgreSQL divergence every 15 minutes
- A mirror count reconciler alerts on drift and never repairs automatically
- On-demand integrity check through the API

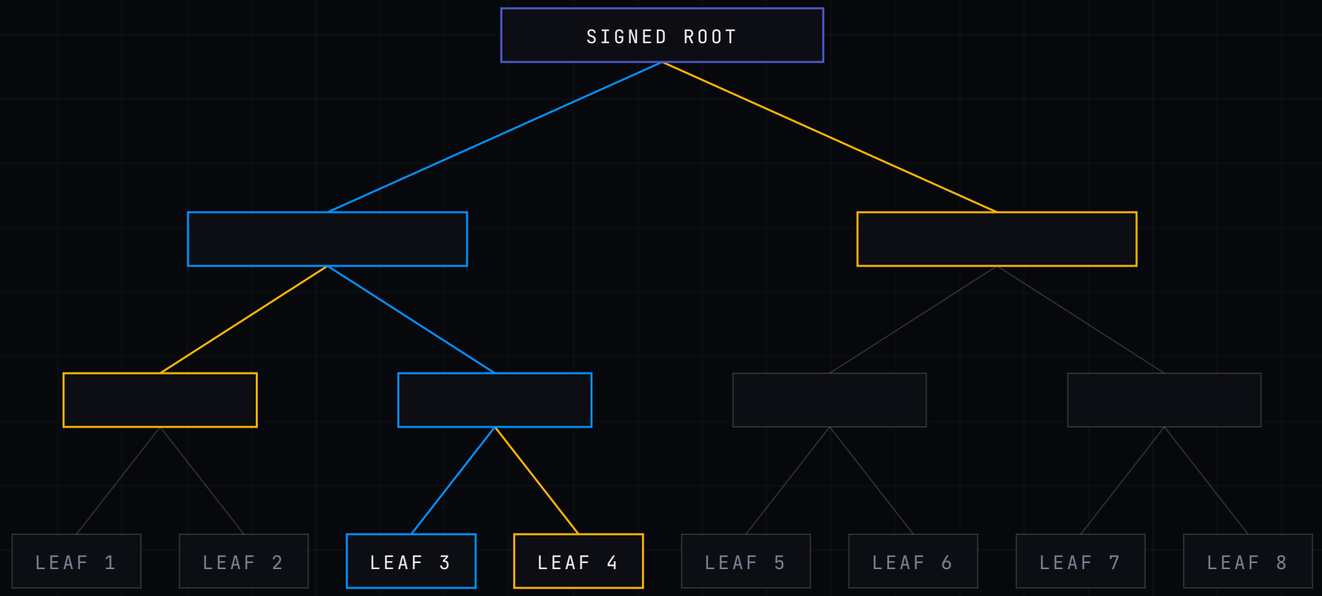
SIGNED CHECKPOINTS

- Every 1,000 entries or 15 minutes, signed and chained to the previous checkpoint
- Record chain length, head hash, evidence head and transparency root
- Verified at boot; failure stops the start
- Optional witnesses and anchors extend trust beyond the gateway

27 TRANSPARENCY LOG, RECEIPTS AND PROOFS

Prove a decision was logged without trusting the *operator.*

Each decision registers a signed leaf in an append-only Merkle log with RFC 6962 inclusion and consistency proofs.



Inclusion proof for leaf 3: the sibling hashes (amber) along the path from leaf 3 (blue) are enough to recompute the signed root.

SIGNED ROOTS AND RECEIPTS

- Roots generated every 60 seconds by default and chained
- A signed receipt is issued only for persisted leaves
- Receipts are held for one hour by default, up to seven days

PUBLIC INTERFACE

Entries, roots, inclusion proofs, consistency proofs and verification are public; key sets are published at a well-known address.

NEVER SILENTLY LOST

- A leaf that cannot be written falls to a recovery queue, then a file journal
- Execution is refused with 503 if the journal stays unavailable beyond 60 seconds

CHECKED FOR GAPS

An optional scanner verifies continuity, hash links, roots and signatures, and classifies any gap as INFO, WARNING or CRITICAL.

28 INDEPENDENT AND OFFLINE VERIFICATION

Check the evidence without QODIQA *running.*

Verifiers import nothing from the product, and trust always comes from a key set published or supplied out of band.

	EVIDENCE	VERIFIED BY	TRUST ANCHOR
01	**Signed artifact**	SDK helpers, standalone verifiers, the verify API	Published public keys
02	**Evidence package and bundle**	Standalone package and bundle verifiers	Key file or pin supplied out of band; keys inside a bundle are never trusted
03	**Audit chain and checkpoints**	Independent chain kit that imports no product code	Checkpoint key set
04	**Transparency entry, root and proof**	SDK verification and the independent kit	Transparency key set
05	**Receipt**	SDK and the independent kit	Transparency key set
06	**External anchor**	Independent kit in anchor-trust mode	Customer-held anchor
07	**Full request proof**	Proof assembler bundle, checked offline	Published key sets
08	**Any submitted artifact**	Verification portal with eight checks	Published public keys

ARTIFACT VERIFICATION LADDER

- Structure
- Canonicalisation
- Key resolution: active or deprecated only
- Timestamps
- Signature
- Optional inclusion proof

PROOF ASSEMBLER

One tenant-scoped `qodiqa-proof` document per request gathers audit entries, chain path, leaf, proof, receipt, anchors and keys.

29 PROTOCOL V2, CANONICALISATION AND THE MAC KEYRING

Exact bytes in. Tamper-evident state *out*.

Protocol V2 binds more of the request under a strict canonical form; a MAC keyring protects internal state at rest.

PROPERTY	PROTOCOL V1	PROTOCOL V2
CANONICAL FORM	Stable-stringified JSON	JCS-P1, a strict profile of RFC 8785
BINDS	Method, path and body	Method, path, query, declared headers and body
ISSUANCE	Default	Opt-in by configuration
ACCEPTANCE	Default	Opt-in by configuration
SIZE AT ISSUANCE	Request limits	64 KiB canonical
MINIMUM VERSION	Global floor, default 1; tenant minimum 1 or 2	Tenants can require V2

JCS-P1 REFUSES

- BYTE-ORDER MARKS
- INVALID UTF-8
- DUPLICATE KEYS
- OUT-OF-RANGE NUMBERS
- NON-NFC STRINGS
- LONE SURROGATES
- EXCESSIVE DEPTH
- OVERSIZE INPUT

MAC KEYRING

An HMAC-SHA256 keyring seals nine kinds of internal state so that a shared cache or a stored record cannot be forged or downgraded.

- CONSENT CACHE ENTRIES
- REVOCATION EVENTS
- ORGANISATION STATUS
- ORGANISATION STATUS EVENTS
- CONTROL-PLANE NOTICES
- KILL SWITCH
- PROVIDER REGISTRY
- OPERATOR SESSIONS
- SAML LOGIN STATE

– Keys encrypted at rest under the key-encryption key; a fingerprint guard rejects a wrong pairing
– Verification accepts the active and previous key, so rotation does not invalidate recent tags
– Rotation by operator command or the key administration interface, with step-up
– In production the keyring must already exist; a missing keyring is a fatal start error

A reserved `qodiqa` claim block carries version, kind (request or action), binding, optional model and limits. Callers cannot inject that name. Unknown critical claims are refused. Action binding and signed receipts use the same canonicaliser. A normative specification and generated vectors are checked against the code.

30 TRANSPARENCY GAP DETECTION, WITNESSES AND EXTERNAL ANCHORS

Find gaps. Anchor the record outside the *gateway*.

A scanner checks the transparency log for breaks, and checkpoints can be co-signed and anchored outside the gateway.

Audit chain 01
Hash-chained entries with transparency references

→

Signed checkpoint 02
Every 1,000 entries or 15 minutes; Ed25519; chained

→

Witness co-signature 03
None by default; local key or operator program; refuses forks

FILE EXPORT	Append-only anchor file held by the customer
GIT COMMIT	Local repository, optionally signed; never pushed
OPENTIMESTAMPS	Via an operator-supplied executable
RFC 3161	Via an operator-supplied TSA executable

GAP SCANNER

Optional, off by default, requires PostgreSQL. Incremental or full scans every five minutes by default.

- ID continuity, hash links and entry-hash recomputation
- Genesis, root chain, root hash and root signature and key status
- Eight gap classes, from hash-chain break to artifact without a log entry
- Classified INFO, WARNING or CRITICAL, alerted and counted

WRITE-PATH DURABILITY

- A leaf that cannot be written falls to a PostgreSQL recovery queue, then a file journal
- Blocks and streams may also use a bounded memory buffer
- A dead-letter queue offers list, replay and archive
- Execution is refused with HTTP 503 if the journal is unavailable beyond 60 seconds by default

HONEST LIMITS

- QODIQA contacts no network witness, timestamp authority, OpenTimestamps calendar or remote git host itself
- Any network contact happens through operator-supplied executables
- A local witness is not independent of the operator
- Anchoring covers checkpoints, which embed the transparency root
- Without anchoring configured, production starts with a warning

31 RETENTION, LEGAL HOLDS AND DATA-SUBJECT RIGHTS

Keep what must be kept. Erase what must be *erased*.

Retention applies to operational events, legal holds suspend it, and data-subject requests run a tracked, evidenced workflow.

Retention policy 01
Per tenant, category, severity and event type

→

Expiry stamping 02
First run about five minutes after start, then every 24 hours

→

Batched deletion 03
Dedicated retention role; each run recorded

→

Legal hold check 04
A hold blocks deletion for its artifact, tenant or category

RETENTION AND HOLDS

- Automatic retention covers the platform operational and security event store
- Enforcement events are never auto-deleted, and the audit chain is untouched by the job
- Holds name a reason and scope to an artifact, tenant or category
- Policies, runs and holds are managed through a stepped-up admin API

PRUNING CHAIN HISTORY

- An operator-only command, never scheduled; off until a retention period is set
- Needs the server stopped and a confirmation flag
- Writes a signed base record and an append-only prune record so the shortened chain still verifies
- A separate archive export seals history first

DATA-SUBJECT REQUESTS

- Types: rectification, erasure, restriction, portability
- States: received, identity verified, approved, in progress, then completed, rejected or failed
- Identity verification is an operator attestation
- Portability exports carry a SHA-256 checksum; only checksum and metadata are stored

HOW ERASURE WORKS

Refused while a restriction applies 01

→

Pseudonymised subject reference 02

→

Consent records tombstoned 03

→

Registered data targets erased 04

→

Caches invalidated by signed broadcast 05

→

Per-subject key destroyed, so pseudonyms become unlinkable 06

The audit chain is never rewritten. Newly written entries carry pseudonymised identities; earlier entries keep the values they were hashed with.

32 AGENTS, TOOLS, ACTIONS AND MCP

Agents act only with signed *authority.*

Every tool call is authorised by default-deny rules, signed as an action artifact, executed once and recorded.

Agent session 01
Gateway-issued identity, proof of possession, action budget

→ **Tool registry** 02
Per-tenant versioned definitions; changes need review

→ **Constraints** 03
Class limits, attribute rules and data-loss scan of arguments

→ **Approval gate** 04
Delegation chain checked; sensitive tools wait for a human

→ **Action artifact** 05
Signed Ed25519, valid 60 seconds, positioned in the session chain

→ **Tool gateway** 06
Re-verifies everything, one-shot, intent and commit recorded

→ **MCP server** 07
HTTPS JSON-RPC with pinned identity and size limits

TOOL RESULTS ARE SCREENED BEFORE THEY RETURN TO THE AGENT

Data-loss scan → **Injection heuristics** → **Semantic gates** → **Untrusted-result wrapper**

NINE ACTION CLASSES

READ · WRITE · FILE · DATABASE · EXTERNAL_API · COMMAND · COMMUNICATION · FINANCIAL · DELEGATE

Approvals

Sensitive actions return a pending request and wait for a human. The approver must be independent of the requester and pass MFA step-up. The approval is a signed, single-use statement.

Delegations

Signed delegation chains that can only narrow authority. Maximum depth of three by default, a lifetime of up to 24 hours, cycle checks, and a revoke cascade that ends dependent sessions.

Boundaries

Agent actions use their own path: authorise, sign, call. They do not go through the provider router. The MCP client speaks HTTPS only; there is no stdio and no process execution. Revocation reaches running calls within about a second.

33 ACTION AUTHORISATION AND AGENT PLANS

Refusal by default. Plans make it *precise.*

Every tool call passes nine ordered checks. An optional plan pre-declares exactly which tools an agent may call, and how often.

Agent and session live 01 →	Proof of possession 02 →	Tool registered 03 →	Constraints and attributes 04 →	Data-loss scan 05 →
Delegation chain 06 →	Approval 07 →	Action plan 08 →	Sign action artifact 09 →	Outcome 201 artifact · 202 pending · BLOCK

AGENT ACTION PLANS

- A plan names steps; each step is a server and tool with a maximum call count
- At most 50 steps, up to 10,000 calls per step, valid for up to 24 hours
- One active plan per session; a new plan closes the previous one
- Usage is charged atomically when the artifact is issued, so concurrent calls cannot exceed the limit

TOOL REGISTRY

- Per-tenant, versioned definitions with a restricted schema
- Action class, reversibility and approval requirement recorded
- New or changed tools enter review; the registrant cannot approve their own
- A schema hash detects change between authorisation and execution

APPROVALS AND DELEGATIONS

- States: pending, approved, denied, expired, cancelled
- Approver independent of requester and agent owner; MFA step-up
- Delegation chains narrow authority, depth three by default, up to 24 hours
- Revoke cascades to dependent sessions

Per-class constraints

File tools are limited by path prefix, database tools by permitted operations, external API tools by host allow-list and command tools by command allow-list. Anything not allowed is denied.

Execution

The tool gateway re-verifies the artifact, session, tool, arguments, approval and delegation, executes once, and records intent and commit entries with a hashed receipt.

PART 04

04

Enterprise *platform.*

Interfaces, consoles, compliance, state, operations and deployment.

35 APIs and developer integration
36 SDKs and client behaviour
37 Tenant portal and operator consoles
38 Attestation, certification and compliance reporting
39 Webhooks and usage metering
40 Data architecture and persistence
41 Backup, restore and disaster recovery
42 Observability and operational intelligence
43 Alerting, incidents and escalation
44 Deployment and integration architecture

35 APIS AND DEVELOPER INTEGRATION

Sign, call, decide, *verify.*

Integration takes a few calls, and the same contract serves applications, administrators and public verifiers.

Obtain an artifact 01

A scoped machine identity asks for a signed artifact bound to the request and the tenant policy set.

SIGN

Make the call 02

The application sends the request with the artifact in a header. Streaming uses the stream route.

CHAT

Receive a decision 03

The response carries the artifact, key and transparency references, plus an optional signed receipt.

EXECUTE OR BLOCK

Verify anywhere 04

SDK helpers and standalone verifiers check artifacts and inclusion proofs without a QODIQA runtime.

OFFLINE OR BY API

CREDENTIALS FOR INTEGRATIONS

- **Signing identities** request artifacts, scoped to a tenant, environment and provider
- **Operator credentials** administer the platform with role-based permissions
- **Public endpoints** for verification, transparency reads and key discovery need none

TWO PLANES, TWO SHAPES

- Machine surfaces (sign, chat, verify, actions, tools) answer with a decision body
- The administrative plane answers with a management envelope and requires step-up for changes
- A disabled plane answers as if it did not exist

WAYS TO INTEGRATE

- Call the API directly
- Use an SDK in JavaScript, Python, Go or Java
- Put middleware in front of a service so it accepts only valid artifacts
- Run the local developer sandbox

ONE GENERATED CONTRACT

- OpenAPI 3.1 generated from the running routes
- A public launch subset is published separately
- SDK operation tables derive from the same contract

36 SDKS AND CLIENT BEHAVIOUR

Four clients. One operation *table.*

The SDKs wrap the same contract, with retries, streaming, step-up and offline verification built in.

JAVASCRIPT AND TYPESCRIPT

- Canonical SDK, version 1.1.0, for Node 18 or later
- Typed resources plus a generic client for the whole API
- Offline verification, webhook verification, agent proof signing, receipt and anchor checks
- Express, Fastify and Connect middleware

PYTHON (BETA)

- Version 1.0.1, in beta
- Synchronous and asynchronous clients with typed resources
- Offline verification, webhook verification, agent proof signing
- FastAPI and Django middleware

GO AND JAVA

- Version 0.1.0 each
- Generic client for the whole API
- Streaming, retries and one-time-password step-up
- Offline artifact and transparency verification

CLIENT BEHAVIOUR

- 30-second timeout and two jittered retries; Retry-After honoured
- POST and PATCH retry only on 429, never on AI chat
- MFA-required responses resend once with a one-time password
- An unterminated stream is reported as an error

DEVELOPER SANDBOX

- `qodiqa dev` runs the real gateway in an isolated temporary environment
- Starts disposable Redis and PostgreSQL in Docker when needed
- A built-in deterministic provider needs no provider account

VERIFICATION AND KEYS

- Remote key discovery with caching
- Webhook signature check with a 300-second tolerance
- Legacy clients are deprecated

37 TENANT PORTAL AND OPERATOR CONSOLES

Self-service for tenants. Consoles for *operators.*

The gateway serves its own browser interfaces. Pages are static and unauthenticated; every datum comes through an authenticated API call.

Login request 01
OIDC authorisation code with PKCE; signed flow state valid five minutes

→

Identity provider 02
Tenant users authenticate at their own provider

→

Callback 03
State and tenant claim verified; MFA assurance enforced when required

→

Portal session 04
HMAC-signed token, eight hours by default, bound to user, tenant and role

→

Read-only portal 05
Identity, keys, consent and usage

	CONSOLE	WHAT IT DOES	ACCESS
01	**Landing page**	Product introduction and health reference	Public page
02	**Execution Studio**	Browser client that signs a request and calls chat with the artifact; the key is held in memory only	Signing-identity credential
03	**Operational dashboard**	Overview, executions, blocks, replays and anomalies with live updates	Operator with dashboard permission
04	**Audit viewer**	Search, timeline, inspect, lineage, export and investigations	Operator with audit permission
05	**Policy management**	Governed policy editor	Operator with policy permission
06	**Approvals**	Human decisions on agent actions	Operator with approval permission
07	**Tenant portal**	Tenant self-service, off by default	Portal session

The tenant portal is off by default and exists only when enabled by configuration. Its read-only interface serves identity, keys, consent, daily token usage and a CSV usage export, and it is separate from the public verification portal.

38 ATTESTATION, CERTIFICATION AND COMPLIANCE REPORTING

Evidence quality, scored. Not a regulatory *verdict.*

Attestations judge authenticity, certification scores evidence quality, and compliance reports map telemetry to controls.

ATTESTATION AND VERIFICATION PORTAL

- **Verification portal:** eight-check verification of an artifact, VERIFIED, VERIFIED WITH WARNINGS or FAILED
- **Attestation service:** records ATTESTED, ATTESTED WITH WARNINGS or NOT ATTESTED for artifacts, evidence packages and chains of up to 50 records
- Stored per tenant under row-level security; anonymous results are ephemeral

EVIDENCE PACKAGES

Per-artifact packages cover artifact, policy, consent, caller, key, verification, audit and replay sections, sealed with Ed25519, exportable as JSON, PDF or a bundle with a Merkle inclusion proof.

CERTIFICATION

- Certification evaluates submitted artifacts and evidence, out of 100
- 95 or more: CERTIFIED PREMIUM; 80: PLUS; 60: CERTIFIED
- Critical failures force NOT CERTIFIED
- A signed certificate artifact is issued, valid 365 days by default
- Batch evaluation, up to five items at a time

SCORE WEIGHTS

Artifact verification	15
Evidence generation	15
Signature	10
Evidence verification	10
Transparency registration	10
Audit coverage	10
Policy compliance	10
Conformance	10
Chain	5
Control coverage	5

COMPLIANCE REPORTING

GDPR · 8 CONTROLS | EU AI ACT · 8 CONTROLS | ISO/IEC 42001 · 9 CONTROLS | NIST AI RMF · 12 CONTROLS

- Evidence is collected, evaluated, scored and optionally signed; export as JSON, HTML or PDF
- Controls are PASS, PARTIAL or NOT APPLICABLE; overall COMPLIANT, PARTIALLY COMPLIANT or NON COMPLIANT
- AI disclosure notice per EU AI Act Article 50 on every AI response

Reports score collected telemetry against rules. They are not independent attestations, certifications or audits, and they do not establish legal compliance. ISO 27001 and SOC 2 appear only as a documentation-level mapping.

39 WEBHOOKS AND USAGE METERING

Signed events out. Usage visible per *tenant.*

Administrators subscribe HTTPS endpoints to platform events, and tenants can see their AI token usage by day.

Event 01
Certification, attestation and approval events

→

Queue 02
Persisted for asynchronous delivery

→

Signed delivery 03
HTTPS POST with an HMAC-SHA256 signature over timestamp and body

→

Retry 04
Up to five attempts, 5 to 300 second back-off

→

Dead letter 05
Kept for replay after final failure

WEBHOOK SECURITY

- HTTPS only, TLS 1.2 or later, certificate verified
- Private, loopback, link-local and metadata addresses blocked; DNS re-validated at connect time; redirects re-validated
- Signing secret shown once and stored encrypted
- Per-host circuit breaker; transient failures retried, permanent ones not

USAGE METERING

- Read-side aggregation of token usage from audit entries
- Daily UTC buckets; default 30 days, up to 400
- Shown in the tenant portal with a CSV export
- No write-time metering or billing in the product

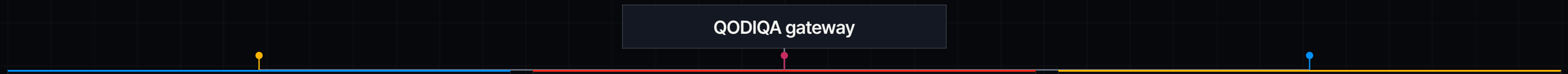
MANAGEMENT

- Tenant-scoped endpoints behind an admin permission and MFA step-up
- Event and delivery history
- Verified in the SDKs with a signature helper

40 DATA ARCHITECTURE AND PERSISTENCE

Three stores. Three distinct *jobs.*

Durable evidence lives in PostgreSQL, live protection state in Redis, and operator-owned material in files.



DURABLE AND TENANT-ISOLATED

PostgreSQL

- Audit mirror and evidence chain
- Transparency log and checkpoints
- Consent, DSR and policy governance
- Control plane and alerting
- Compliance and agent records
- Row-level security per tenant
- Append-only guards on evidence
- Separate roles for app, system, reporting, retention and backup

LIVE STATE, NEVER THE RECORD

Redis

- Replay nonces and anomaly windows
- Rate-limit counters
- Authentication lockout
- Idempotency results
- Consent cache and revocation events
- Transparency receipts
- Key-rotation lock
- Single, Sentinel or Cluster modes
- Persistence verified at start

LOCAL AND OPERATOR-OWNED

File-based stores

- Audit log with write-ahead log
- Encrypted key files
- Policy store
- Organisations, tenants, RBAC users
- Signing identities
- Recovery journal
- Checkpoint sidecars
- Control plane can move to PostgreSQL in production

Tenant isolation in the database

Row-level security is forced on tenant tables and the tenant id is set per transaction, so it cannot leak across pooled connections. An unset tenant returns no rows. Cross-tenant reads go only through pinned, controlled functions.

Operational state

The operational dashboard is aggregated in memory and rebuilt from the audit chain at start. Receipts and idempotency results live in Redis or process memory and expire by design. Redis is never the system of record.

41 BACKUP, RESTORE AND DISASTER RECOVERY

Recoverable by design, verifiable after *restore*.

Backup and restore cover file components and PostgreSQL, and a restore ends with integrity checks rather than hope.

Backup 01
File components and PostgreSQL, snapshot-consistent or portable row dump

→

Seal 02
Encrypted, HMAC-sealed, manifest and checksums, tiered retention

→

Verify 03
Non-destructive verification and restore validation

→

Restore 04
Three-phase, fail-closed, into a clean target that is not the source

→

Repair 05
Ownership repaired, chain heads restored, erasures replayed, probes run

PROTECTION OF BACKUPS

- Encrypted by default (AES-256-CBC with PBKDF2) and sealed with an HMAC
- Daily, weekly and monthly retention tiers
- Optional off-peak window; failure alerts and metrics

POINT-IN-TIME RECOVERY

- Write-ahead-log archiving and point-in-time restore
- Off-site hooks
- Restore targets are guarded so a dump cannot overwrite its source

WHAT IS NOT RESTORED

- Redis state is ephemeral and is not restored
- Erasures recorded in an append-only ledger are replayed after restore so deleted data stays deleted

SCHEDULING

- Kubernetes CronJobs and systemd timers are provided as configuration
- Their deployment is an operator decision

42 OBSERVABILITY AND OPERATIONAL INTELLIGENCE

Every layer emits signals you can *use*.

Logs, metrics, traces, health and dashboards describe the platform without exposing secrets.

STRUCTURED LOGGING

- One JSON line per event, labelled as security, audit or operational
- Sensitive fields and credential-shaped values are redacted on every event
- Noisy failures are sampled and aggregated

METRICS

- A built-in Prometheus registry covering decisions, streaming, the audit chain, the transparency log, stores, keys, protection controls and semantic gates
- Optional push over OTLP

TRACING

- Opt-in, using W3C trace context
- One root span per request with redacted attributes
- Trace identifiers appear in log lines

LOG SHIPPING

- ELK, Datadog, Splunk and generic SIEM or syslog destinations
- ECS, CIM and CEF formats
- Queues are bounded, retried and redacted again

HEALTH AND DIAGNOSTICS

- Liveness, and readiness by component
- Per-store Redis health
- Authenticated diagnostics; opt-in profiling

DASHBOARDS

- A live operational dashboard with executions, blocks, replays and anomalies
- Example Grafana dashboards and Prometheus alert rules are provided

LATENCY MONITORING

- Mean and p99 request duration checked on an interval against thresholds
- Breach and recovery events feed alerting

RESOURCE AWARENESS

- Event-loop delay monitor
- Bounded queues, pools and buffers
- Readiness reflects saturation and backlog

43 ALERTING, INCIDENTS AND ESCALATION

From a single event to the right *person*.

The alerting engine turns security and operational events into scored, deduplicated alerts that reach people through the channels they use.

Detect 01
Five detectors react to events

→

Score 02
Risk 0 to 100, severity INFO to CRITICAL

→

Deduplicate 03
Suppress by rule; store in PostgreSQL

→

Correlate 04
Related alerts become incidents

→

Notify 05
Seven channels with retry and rate limits

→

Escalate 06
Timed levels up to four

WHAT IT DETECTS

- Replayed nonces, key expiry and mismatch, invariant violations
- Policy load failure, Redis and PostgreSQL degradation
- Audit-chain corruption, mirror drift, signing-provider failure
- Latency breach, disk full, consent-propagation failure
- RBAC denials, operations-auth failures, sensitive data in output
- Volume, block-rate and error-rate anomalies per tenant and key

CHANNELS

WEBHOOK · SLACK · MICROSOFT TEAMS · EMAIL · SMS · SIEM · PAGERDUTY

- Each channel is enabled when its recipient is configured
- Per-channel rate limits and retry with back-off
- SMS is reserved for critical alerts

OPERATING THE QUEUE

- Admin API to list, acknowledge, resolve, suppress and mark false positives
- Live alert stream and channel health
- Retention by severity, 30 to 730 days
- Escalation timers are held in memory

44 DEPLOYMENT AND INTEGRATION ARCHITECTURE

From one hardened image to a full *cluster.*

Deployment mechanisms ship with the product and are validated against the same production profile the gateway enforces at boot.

01

Container image

- Two-stage build on a pinned Node 22 base
- Non-root user, read-only code
- No package managers
- Build fails if keys or environment files are present

02

Docker Compose

- Gateway, Redis and PostgreSQL
- Redis with mutual TLS
- PostgreSQL over TLS
- Read-only filesystem
- All capabilities dropped
- Seccomp profile

03

Kubernetes

- Kustomize base and overlays
- Single-node StatefulSet
- Default-deny network policy
- Backup and recovery CronJobs
- Examples for HSM, Vault, External Secrets and ingress

04

Helm and Terraform

- Helm chart with migration job, ingress, autoscaling and probes
- Monitoring rules
- Terraform module wraps the chart and defines no cloud resources

05

Operations

- Backup and recovery CronJobs and systemd timers
- Health and readiness probes
- Graceful shutdown drain
- Production boot validation
- Alerting and dashboards

TOPOLOGY

- The audit chain is single-writer by design, so a deployment runs one active gateway process
- It is backed by PostgreSQL for durable records and Redis for live protection state
- Shipped manifests describe the intended operating model; they are mechanisms, not a claim about any site

INTEGRATION BOUNDARIES

- Key custody, identity providers, AI providers, observability and notification systems are optional external services
- Each is reached through a defined interface and enabled by configuration

PART 05

05

The complete *platform.*

The whole capability map, a worked scenario, the integration model and why the combination matters.

46 Capability map
47 An illustrative execution scenario
48 Integration and interoperability model
49 Why the architecture matters

46 CAPABILITY MAP

The whole platform, grouped by *purpose.*

Ten capability groups. Each is explained once in this document and linked to the others through the execution path.

01

Gateway and API

- Hardened HTTP edge
- Generated OpenAPI contract
- Idempotency keys
- Admission control
- Kill switch
- Streaming over SSE

02

Consent and policy

- Consent lifecycle
- Signed revocation events
- 11 policy rule types
- Tenant floor
- Maker-checker governance
- Visual policy builder

03

Semantic inspection

- 13 rule classifiers
- Five embedding gates
- Intent gate
- Output inspection
- DLP with 24 detectors
- Content rules

04

Execution

- Eight provider families
- Runtime compatible providers
- Policy-driven routing
- Breakers and failover
- Execution invariant
- Budgets

05

Identity and access

- Six RBAC roles
- OIDC, SAML, LDAP
- TOTP MFA step-up
- Signing identities
- Organisations and tenants
- Authentication lockout

06

Cryptography

- Ed25519 artifacts
- Domain separation
- Key rotation
- KMS and HSM custody
- Execution tokens
- Field encryption

07

Evidence

- Hash-chained audit log
- Signed checkpoints
- Evidence ledger
- Merkle transparency log
- Signed receipts
- Witness and anchors

08

Agents and tools

- Agent sessions
- Tool registry
- Human approvals
- Delegation chains
- MCP gateway
- Result screening

09

Privacy and compliance

- Data-subject requests
- Erasure workflow
- Compliance reports
- AI disclosure notice
- Verification portal
- Usage metering

10

Operations and developers

- Metrics, logs, traces
- Seven alert channels
- Backup and recovery
- Four SDK families
- Offline verifiers
- Local sandbox command

47 AN ILLUSTRATIVE EXECUTION SCENARIO

One request, end to *end*.

The documented capabilities working together on a permitted request, and on the same request when it is refused.

ILLUSTRATIVE SCENARIO · A SUPPORT ASSISTANT SUMMARISES A TICKET

01 **Artifact issued**
Scoped identity obtains a signed artifact

02 **Request sent**
Prompt and artifact reach the gateway

03 **Artifact verified**
Signature, binding and one-time nonce

04 **Consent and policy**
Consent active, rules pass

05 **Inspection**
Prompt screened; an email address is redacted

06 **Intent recorded**
Written before the provider is called

07 **Provider call**
Router dispatches with an execution token

08 **Response checked**
Output inspected, then released

09 **Evidence**
Audit entry, log leaf and receipt

PERMITTED PATH — The response returns with artifact, key and transparency references, and an optional signed receipt.

VARIATION · THE SAME REQUEST WITH AN INJECTED INSTRUCTION

Artifact issued — **Request sent** — **Artifact verified** — **Consent and policy** — **Inspection** — BLOCK · REASON CODE RETURNED

The provider is never called. A block entry and transparency leaf are still written, so the refusal can be verified later.

LATER

An independent party verifies
Using published public keys and a standalone verifier, with no access to the gateway.

An operator exports evidence
An evidence bundle joins the record to its transparency inclusion proof.

The chain is checked continuously
A live verifier walks the audit chain and reports any break; it never repairs.

This scenario is a composite illustration of documented behaviour. It does not describe a real customer or deployment.

48 INTEGRATION AND INTEROPERABILITY MODEL

One gateway. Eight surfaces around *it.*

Every external system meets QODIQA at a defined interface: standard HTTPS, signed artifacts, and pluggable providers.

Applications and SDKs
JavaScript · Python · Go · Java
Express, Fastify, FastAPI, Django middleware
Standard HTTPS and an OpenAPI 3.1 contract

Identity providers
OIDC tokens · SAML 2.0 · LDAP and Active Directory
Operator sessions and machine identities
MFA assurance honoured for step-up

AI providers
OpenAI · Anthropic · Google Gemini · Mistral
Azure OpenAI · Amazon Bedrock · Llama · local
Runtime-registered OpenAI-compatible endpoints

Operators and auditors
Console, policy builder and audit viewer
Admin API behind RBAC and MFA step-up
Verification portal and compliance reports

GATEWAY

QODIQA gateway

Enforce · sign · execute · record

ONE REQUEST PATH, ONE EVIDENCE TRAIL

MCP servers and tools
HTTPS JSON-RPC with pinned server identity
Tools registered, versioned and reviewed
Actions signed, executed once and recorded

Key custody
Local encrypted keys · HashiCorp Vault
AWS KMS · Azure Key Vault · Google Cloud KMS
PKCS#11 hardware security modules

Data stores
PostgreSQL for durable records
Redis for replay, limits and revocation
File-based stores for keys and policies

Observability and notification
Prometheus · Grafana · OTLP · ELK · Datadog · Splunk
Slack · Teams · email · SMS · PagerDuty
Signed webhooks

49 WHY THE ARCHITECTURE MATTERS

The combination is the *point.*

Each capability is useful alone. Together they make enforcement exact, decisions provable and failures visible.

01	**Controls on the only path**	Safeguards beside a model can be bypassed. QODIQA places them on the one route to a provider and refuses dispatch without a verified context.
02	**Authorisation that travels**	A signed artifact binds the exact request and policy set, so what is executed is what was approved.
03	**Failure becomes refusal**	Dependency loss, missing state and engine errors produce a BLOCK decision, not an unrecorded execution.
04	**Judgement informs, rules decide**	Semantic scores assess meaning; deterministic checks and signatures decide. Neither pretends to be the other.
05	**Evidence outlives the request**	Chains, checkpoints and Merkle proofs let third parties verify what was decided, with out-of-band trust anchors.

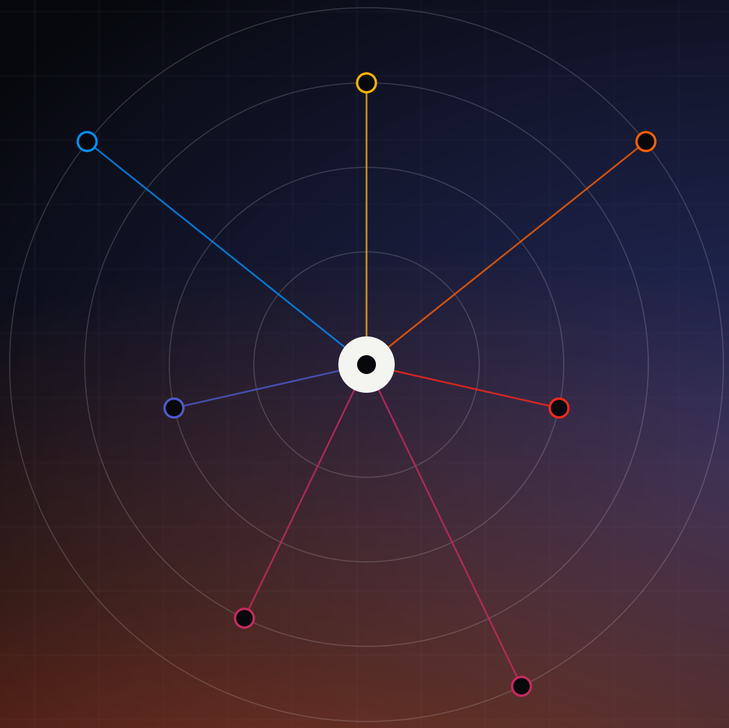
QODIQA provides evidence about enforcement. It does not claim that AI outputs are correct, nor does it determine the legal admissibility of any record.

50 IN SUMMARY

QODIQA

Control what AI executes. Prove *it*.

QODIQA places verification, consent, policy, inspection and signed evidence on the execution path between applications and AI, so that every consequential request is decided before it runs and can be checked after it has.



CONTACT
contact@qodiqa.com

WEB
www.qodiqa.com

DATE
October 2026