OWLIE

SECURITY WHITEPAPER — SW-01

Security by design

How Owlie protects identities, access decisions, and the credentials that connect it to your systems.

VERSION	DATE	CLASS	PREPARED BY
1.0	2026-09	PUBLIC	SECURITY ENGINEERING

Contents

HOW TO READ THIS PAPER

Sections 01–10 present the security controls relevant to customer evaluation. The technical appendix describes protocol behavior, execution restrictions and implementation-specific boundaries.

01 Executive security posture

§ 01
OVERVIEW

Owlie manages identities and access across connected systems. It evaluates access policies, routes requests for approval, runs access reviews and provisions the resulting changes. Its connections to those systems give it access to sensitive records and credentials.

To protect that access, Owlie checks tenant identity and permissions on requests, requires recent authentication for sensitive changes, and encrypts selected secrets with tenant-specific keys. Customer code runs with restricted access to the network and host credentials.

This paper describes how these controls work, where they apply, and which settings customers manage. It also covers the infrastructure protections supplied by Cloudflare and PlanetScale.

INDEPENDENT ASSURANCE

Owlie has completed a SOC 2 Type II examination. The report is available under NDA through the Owlie Trust Center. Section 9 describes the reporting period and the status of independent penetration testing.

The main paper presents the security controls relevant to customer evaluation. The technical appendix in this same document describes protocol behavior, execution restrictions and implementation-specific boundaries.

02 Architecture and tenant isolation

§ 02
ARCHITECTURE

Owlie is a multi-tenant application built on Cloudflare Workers, with PostgreSQL as its relational database. Separate services handle authentication, application APIs, encryption, function execution and identity-governance workflows. Public request handling identifies the tenant and actor before the main API invokes domain logic.

HOSTING +
DATA LOCATION

Hosting and data location

Cloudflare operates the underlying compute platform and isolates Worker execution using its runtime security model. Owlie uses service bindings for internal service calls, Durable Objects for stateful coordination, and R2 for object storage.

The production relational database is PlanetScale Postgres, hosted on AWS in Northern Virginia (`us-east-1`), with two replicas. Production and staging use separate database branches and Hyperdrive configurations. Application database access uses Cloudflare Hyperdrive, with query-result caching disabled. PlanetScale requires encrypted database connections and encrypts database storage and backups at rest.

Cloudflare automatically encrypts R2 objects and metadata at rest using AES-256 with Cloudflare-managed keys. Durable Object storage is also encrypted at rest with Cloudflare-managed keys. Owlie's application-level encryption of selected secrets, described in section 4, adds a separate layer of protection to these storage controls.

LAYER	PROVIDER	PROTECTION AT REST
Application compute	Cloudflare Workers	Runtime isolation model
Relational database	PlanetScale Postgres	Encrypted storage + backups
Object storage	Cloudflare R2	AES-256, managed keys
Stateful coordination	Durable Objects	Encrypted at rest

§ 02 · CONTINUED

Application processing and other storage use Cloudflare's distributed infrastructure, so the database region is not a residency designation for all customer data.

TENANT ISOLATION

Tenant isolation

The request hostname identifies the tenant. Owlie validates the hostname structure and rejects a main-API request when the authenticated identity belongs to a different tenant. Shared service queries apply tenant predicates alongside customer-supplied filters; an OR expression in a filter does not replace the surrounding tenant restriction.

Customers share database and compute infrastructure. Owlie enforces logical tenant isolation in the application: the shared service layer scopes queries to the tenant, and callers of lower-level database operations must supply that scope. Query values are parameterized, and public service filters are checked against permitted fields.

TERMINATION + DELETION

Account termination and data deletion

On account termination, Owlie disables tenant functionality and retains tenant data for `30 days`. After that period, tenant data is deleted from active systems. A minimal deletion receipt remains, recording that the tenant existed and who initiated deletion.

Backup copies expire under the retention schedules described in section 8. When a backup is restored, Owlie uses deletion receipts to remove data belonging to deleted tenants from the restored database.

03 Authentication and authorization

§ 03
SIGN-IN + CREDENTIALS

Sign-in and credentials

Owlie supports password authentication, external OpenID Connect and SAML sign-in, TOTP and WebAuthn passkeys. Browser Secure Password sign-in uses OPAQUE to authenticate without including the password in the login payload. Other password-handling flows have different boundaries, described in appendix A.

Sensitive credential and login-policy changes require recent authentication. Applicable policies reject API-key and impersonated actors, and backup codes do not satisfy step-up authentication. Emergency administrator recovery restores local credentials through a restricted, one-time-code flow; recovery does not itself issue a session.

Session credentials and API keys are validated against tenant and identity state. Session cookies use Secure, HttpOnly and SameSite protections. API keys belong to service accounts and have configurable expiration. Customers manage their permissions, expiry and rotation. Appendix H describes session limits, token rotation and connection revocation behavior.

AUTHORIZATION

Authorization

Authentication establishes the actor; authorization determines what that actor may do. Owlie's central authorization evaluator combines role permissions, tenant- and object-specific grants, and selected relationship rules. Ordinary actors without a matching grant are denied. Classified API operations apply their authorization checks before running the resolver pipeline. Self-scoped operations also rely on domain checks.

Internal services trust calls received through service bindings. System work can use an explicit privileged context. The key-management service also requires callers to enroll and obtain permission for the requested operations, as described in section 4 and appendix B.

§ 03 · CONTINUED

For the main file-download path, access is checked against both the tenant-local file and its owning entity. Responses use attachment handling and private, non-cacheable delivery. Knowing a file identifier alone does not grant access to its contents.

ACCESS GOVERNANCE

Access governance

Deny policies take precedence over grants in the desired-access calculation. Requests and policy exceptions have self-approval restrictions. Access reviews exclude the subject from certification, and reviewer resolution fails closed when no eligible fallback remains. Separate permissions govern policy authoring and activation, subject to the rollback exception described in appendix F.

Owlie checks requested access against policy before starting fulfillment. A blocking deny requires an authorized exception. If a complete policy decision cannot be obtained, fulfillment is held and retried; no provisioning operation or bundle expansion starts from an unavailable or incomplete decision. Appendix F describes the recovery boundary and the remaining approval controls.

Blast-radius checks hold policy decisions whose projected revocations exceed the configured threshold. The default offboarding policy denies access to terminated identities, but administrators can edit or disable it. Termination also disables the person's Owlie account in the same database transaction; connected-system revocation completes separately through provisioning.

04 Encryption and secrets

§ 04
ENVELOPE
ENCRYPTION

Owlie encrypts selected application secrets with tenant-specific envelope encryption. Each encryption uses a fresh 256-bit data key and AES-GCM. A versioned tenant key wraps the data key, and an environment root key protected by AWS KMS protects the tenant keys. Ciphertext authentication binds tenant identity and declared usage context.

KEY SERVICE ACCESS

The key-management service requires caller enrollment and tenant-scoped capabilities with service-specific permissions.

Decryption requests carry a structured reason, and successful cryptographic operations record audit metadata. Business authorization remains with the requesting application service.

ROTATION AND SCOPE

Key rotation selects a new version for new encryptions while preserving the ability to decrypt older ciphertext. It does not automatically re-encrypt historical records.

These controls cover secrets stored through the key-management service; other fields use the infrastructure encryption described in section 2.

CUSTODY

Customers do not have exclusive custody of the application encryption keys.

Appendix B details the key hierarchy, context checks and service enrollment.

05 Connector security

§ 05
CLOUD
EXECUTION

Connectors read accounts and entitlements and apply access changes to connected systems. They run either in Owlie's cloud or on a customer-operated gateway.

Cloud-stored connector credentials use tenant-specific encryption and are decrypted by authorized runtime services when needed. Integration display and editing return secret markers and limited previews rather than complete stored secrets. Cloud HTTP destinations require HTTPS. The shared first-party HTTP client rejects cross-origin redirects and pagination links before forwarding authenticated requests.

GATEWAY
EXECUTION

For gateway-bound integrations, configured secrets reference files or environment variables on the customer's host and are resolved locally. Gateway enrollment uses a short-lived, single-use token. The gateway generates its signing key locally and must prove possession before receiving work.

A custom connector needs both a cloud-signed execution grant and authorization from the gateway operator's local allowlist. The grant identifies the exact connector version and bundle hash, which the gateway checks before loading the code. Custom workers run with restricted filesystem, environment, network and subprocess permissions. The gateway host remains privileged, and first-party connectors can have different permissions.

PUBLISHING +
UPDATES

Published connector versions are immutable. Authoring and publishing have distinct permissions, although one person may hold both. Gateway updates verify signed artifacts and enforce an anti-rollback floor; customers control installation, and automatic updates are off by default.

Appendices C and E cover gateway authentication, update verification, redirect handling and pagination. Appendix J describes synchronization safeguards.

06 AI security

§ 06
AUTHORITY + EXECUTION

Dashboard AI operations use the signed-in user's session. External MCP operations use the caller's service-account key. Host-side tools present those credentials to Owlie's API, where normal API authorization applies.

AI-authored code runs without direct networking or access to the host's SDK credentials. API work is dispatched through the host connector, with execution and call-count limits. The hosted inference provider receives the conversation context supplied by Owlie, including user messages and data returned by authorized tool calls. This can include customer identity and access information relevant to the request.

PROVIDER DATA HANDLING

Owlie disables prompt and response storage and training-data sharing in its Together AI organization. Under Together's zero-data-retention policy, inference content is not persisted or used for training; operational metadata such as token counts and timestamps is retained. For models hosted by DigitalOcean, DigitalOcean's policy states that inputs and outputs are not stored or used for training, and are not sent to the original model creator. These provider controls do not determine how long Owlie retains conversation history.

CONFIRMATION + SECRETS

The dashboard adds confirmation for designated sensitive operations, with durable approval handling for code-mode mutations. MCP uses the service account's authority without that dashboard confirmation requirement. The current actor supplies the confirmation; a second person's approval is not required.

For supported secure-input cards, the browser sends secret values directly to the API and returns receipts without the values to the assistant. The entered secret stays out of the assistant's conversation.

Tenant-authored functions execute in isolated Workers with verified build artifacts and mediated outbound access. Configured secrets are decrypted only for their intended execution context, but are then available to the customer code. Code review and destination approvals therefore remain important customer controls. Appendix D describes the sandbox and the scope of secret redaction.

07 SDLC and vulnerability management

§ 07
DEVELOPMENT
CONTROLS

Development controls

Owlie's dependency policy applies a three-day release-age gate with explicit exceptions. Major upgrades must be selected in the update tool. Dependency build scripts require an explicit allow-or-decline decision. Principal test jobs use a frozen lockfile with install scripts disabled.

CI configuration includes type checking, configuration guards, test discovery coverage and test execution. Additional workflows schedule dependency auditing and AI-assisted source security review. Dependency-audit reporting distinguishes an unavailable registry from a completed scan with no findings, and third-party workflow actions are pinned to immutable commit revisions.

REMEDIATION

VULNERABILITY REMEDIATION

Owlie targets remediation of identified vulnerabilities within `30 days`, regardless of severity.

PRODUCTION
ACCESS

Production access

Production access is restricted to authorized personnel based on operational need. Access approval, review and removal are the responsibility of the production owner. MFA is required wherever available.

§ 07 · INCIDENT RESPONSE

Incident response

Service health is monitored through Axiom, Cloudflare's platform monitoring and Owlie's custom monitors, which check all application services. Alerts are delivered to a Slack channel.

Owlie maintains a documented incident response plan covering reporting, severity assessment, escalation, technical investigation and recovery. The plan assigns security coordination and engineering responsibilities and defines how external communications and notification obligations are assessed. Customer notifications follow applicable contractual and legal obligations. Suspected security incidents can be reported to security@owlie.com.

AUDIT RECORDS

Audit records

Owlie records named security and governance events with actor and tenant context. Audit-log reads require a dedicated permission. Where an operation and its audit insert share a database transaction, they commit or fail together. Standalone audit emission is best-effort and reports failures in operational logs.

Audit retention is tenant-configurable and defaults to `365 days`. Account termination follows the deletion process in section 2. Audit records and access-review event history have different integrity protections, described in appendices G and I.

08 Resilience and disaster recovery

§ 08
DEPENDENCIES

Cloudflare supplies the managed compute and storage platform, while PlanetScale operates the production PostgreSQL database. Owlie maintains the application configuration and a separate encrypted database-backup pipeline. Database recovery and the availability of those providers are distinct dependencies.

OBJECTIVES

RECOVERY OBJECTIVES

24 h	5 min
RTO — operational target to restore service.	RPO for the production database via PlanetScale point-in-time recovery.

These are operational targets, not contractual service guarantees.

MANAGED
BACKUPS

The production database has PlanetScale-managed backups every twelve hours with two-day retention. PlanetScale's point-in-time recovery uses archived transaction logs to restore to a timestamp between the oldest retained backup and approximately five minutes before the present. Managed backups and transaction logs remain in the database's cloud provider and region.

§ 08 · OWLIE BACKUP PIPELINE

Owlie also schedules a daily PostgreSQL dump, encrypts it before upload and stores the encrypted artifact in R2, with deletion configured after 30 days. These daily backups provide older recovery points at approximately 24-hour intervals, rather than extending the five-minute RPO beyond the point-in-time recovery window. The backup path does not need the private decryption key, and the restore tooling can retrieve recovery material from 1Password. Appendix K describes this pipeline and its restore procedure.

Restore keys are held in a restricted 1Password vault accessible to authorized recovery personnel. Recovery access is not dependent on Owlie being available.

RESTORE TESTING

Owlie uses scripted database recovery procedures. Restore tests completed on `2026-09-15` successfully recovered both PlanetScale backups and encrypted R2 backups, with restore execution taking minutes. These tests exercised database restoration, not a complete service outage and recovery.

WORKFLOW DURABILITY

Notification and provisioning workflows use bounded retries and explicit failure states. Stalled sync work is detected through leases, heartbeats and cleanup. Ambiguous external side effects are held for reconciliation rather than repeated without regard to whether they are safe to retry. Provider outages and provider-side processing times can delay fulfillment and revocation.

09 Compliance and assurance

§ 09
SOC 2
TYPE II

Owlie has completed a SOC 2 Type II examination covering April 16 - July 16, 2026. Customers can obtain the report under NDA through the Owlie Trust Center.

EXAMINATION PERIOD	PENETRATION TESTING
2026-04-16 → 2026-07-16	PLANNED — NOT YET COMPLETED

TESTING +
CONTACT

Owlie plans annual independent penetration testing. The first test is planned for later in 2026 and has not yet been completed.

The technical controls described in this paper and the independent assurance report serve different purposes. Customers should consult the report for the examination scope and findings. Security questions and suspected incidents can be reported to security@owlie.com.

10 Shared responsibility

Owlie operates the application and its security controls. Customers manage their tenant's access policies, connected-system permissions and any gateway hosts they operate. The table below summarizes those operational responsibilities; the preceding sections describe each control's scope.

AREA	OWLIE CONTROLS	CUSTOMER RESPONSIBILITIES
Authentication and service accounts	Tenant-bound authentication, additional factors, recent-authentication checks for sensitive changes, and API-key status and expiry validation.	Configure sign-in to match organizational policy; limit service-account permissions and manage key expiry, rotation and revocation. API-key expiry is optional.
Access governance	Policy evaluation, approval workflows, self-approval restrictions and access reviews. New fulfillment is held when a complete policy decision is unavailable.	Maintain policies, approvers and exceptions; review changes to the editable default offboarding policy and redrive a visibly held request if its bounded automatic policy retries exhaust.
Connectors and customer code	Cloud-secret encryption, scoped execution controls and separate authoring and publishing permissions.	Review code, destinations and provider permissions. Assign authoring and publishing roles to different people where separation is required; the product does not require different holders.
AI-assisted operations	API authorization under the current user or service account, with confirmation for designated dashboard operations.	Limit the actor's authority and assess which data is appropriate for AI processing. Dashboard confirmation is not a second-person approval and does not apply to MCP calls.
Connector gateways	Authenticated connections, signed execution grants and verified update artifacts.	Secure the host and local secrets, maintain the connector allowlist, and install gateway updates. Automatic updates are off by default.

Customers should review the permissions and data handling of connected identity providers and external systems alongside their Owlie configuration.

Technical appendix

Protocol behavior, execution restrictions and implementation-specific boundaries behind the controls described in sections 01–10.

A Password protocols and authentication checks

APX A
OPAQUE +
FEDERATION

OPAQUE, password hashing and federation

Owlie supports two password mechanisms. In browser Secure Password sign-in, OPAQUE exchanges authenticate the user without placing the password in the login payload. The implementation uses the P-256 suite. Conventional password authentication uses salted scrypt hashing. Administrative and automation flows may still handle plaintext passwords; the OPAQUE protection applies to browser sign-in.

External OpenID Connect sign-in validates provider signatures, issuer, audience and nonce, with allowed-domain checks. Inbound SAML sign-in checks signed assertion coverage, correlates the authentication request and binds completion to the browser flow. Federated identity trust also depends on the tenant's provider and account-linking configuration.

FACTORS +
RECOVERY

Additional factors and recovery

Owlie supports TOTP and WebAuthn passkeys as additional factors. TOTP secrets are encrypted, and an accepted time step cannot be reused for the same factor across login and step-up flows. Passkey assertion verification checks the expected challenge, origin, relying party, tenant and identity against the registered public-key credential. Challenge consumption is atomic, so concurrent submissions cannot turn the same challenge into multiple successful authentications. MFA backup codes are stored as hashes and consumed with an atomic unused-state check.

Sensitive credential and login-policy changes have additional recent-authentication requirements. The applicable policies reject API-key and impersonated actors, and backup codes do not satisfy step-up authentication. An existing session alone is insufficient for these changes.

Emergency administrator recovery is a separate, restricted flow. One-time recovery codes are bound to an active human administrator or owner, and successful recovery restores local credentials before returning the user to normal sign-in. Recovery does not itself issue a session.

B Key hierarchy and envelope encryption

APX B
TENANT KEYS +
CONTEXT

Tenant keys and context binding

Owlie's key-management service encrypts selected application secrets. Each encryption generates a fresh 256-bit data key and uses AES-GCM for the payload. A versioned tenant key wraps the data key. An environment root key, protected by AWS KMS, protects the tenant keys. Root-key initialization fails closed when the required AWS key specification cannot be verified.

Each tenant has distinct key material. Ciphertext authentication includes tenant identity, data-key identity and declared usage context. For function secrets, the consuming service also checks the expected function and secret name before releasing the value for execution.

ENROLLMENT +
ROTATION

Service enrollment, key access and rotation

Internal callers must prove KMS enrollment using ML-DSA signatures with time and nonce checks before receiving a tenant-scoped capability. That capability preserves the verified service and tenant identity and applies service-specific operation permissions. Business authorization remains with the requesting application service.

Decryption requests require a structured reason, and successful cryptographic operations record audit metadata. Key rotation advances the version used for new encryptions while retaining older versions for decryption. Historical ciphertext stays under its original key version until explicitly re-encrypted.

This encryption covers secrets stored through the KMS service. Other application fields use the storage protections described in section 2. Customers do not have exclusive custody of the application encryption keys.

C Gateway enrollment and execution

APX C LOCAL SECRETS

For gateway-bound integrations, configured secrets are references to environment variables or files on the customer's host. The integration write path rejects literal secret values for those fields. The gateway resolves the references locally when the connector needs them; its error redactor scrubs recognized secret values before reporting errors to the cloud. The customer is responsible for the host and code that can access the resolved secrets.

ENROLLMENT + PROOF OF POSSESSION

Gateway enrollment uses a short-lived, single-use token bound to the tenant hostname. The gateway generates its ML-DSA-65 signing key locally and registers the public key. Each connection must prove possession of the private key before the socket can receive work. Until then, the socket cannot displace an authenticated connection and closes after 15 seconds.

Assertion nonces are single-use within a 120-second freshness window and consumed only after signature verification. Connection and bundle download assertions use different signing domains. Revoking a gateway rejects subsequent handshakes and triggers closure of its active socket.

15 s	120 s	ML-DSA-65
Unauthenticated socket close	Assertion freshness window	Gateway signing key

APX C · EXECUTION GRANTS

Executing a custom connector through the gateway requires both a cloud-signed execution grant and the operator's local allowlist. The grant binds the tenant, gateway, integration, connector version and exact bundle hash. The gateway verifies the bundle against that hash before loading it. The local allowlist lets the operator restrict which connectors the managed gateway will run, even when the cloud authorizes dispatch.

WORKER PERMISSIONS

Custom connector workers start with deny-by-default permissions: no environment-variable access, no filesystem writes or foreign-function interface, and read access limited to the verified bundle. Network permissions are enforced by the runtime, covering raw sockets as well as HTTP. Subprocess permission is denied by default and can be granted per connector. These restrictions apply to custom workers. The gateway host process remains privileged, and first-party connectors may have different permissions.

UPDATE VERIFICATION

Gateway updates verify signed release manifests and artifacts against pinned public keys, check hashes and sizes, enforce an anti-rollback floor, and check the new binary before cutting over. Automatic updates are off by default. The binary trusts only compiled-in public keys; a production build without injected keys fails rather than falling back to development keys.

D Function sandbox and secret handling

APX D ARTIFACT VERIFICATION

Owlie runs tenant-authored functions in dynamically loaded Workers with explicit outbound mediation. Before execution, it checks the tenant-scoped function version, manifest and build linkage, bundle size and SHA-256 digest to verify that the executable matches the recorded artifact. Customers remain responsible for reviewing their code.

MEDIATED EGRESS

Function networking uses HTTPS and configured hostname rules, with destination checks repeated on redirects and a five-hop limit. The tenant's Owlie hostname remains available for the injected SDK. The customer controls the function and its approved destinations, while the runtime mediates network access.

The redirect handler does not forward Authorization or Cookie headers across origins, even when both hosts are approved destinations.

SECRETS + REDACTION

Function secrets are encrypted through KMS and checked against their expected context before injection. Once provided to the function, a secret is available to that code. Administrators should therefore treat code changes, destination approvals and debug execution as security-sensitive choices.

Known configured secret values are redacted from persisted debug request and response payloads, execution logs and errors. Redaction matches known values and may miss transformed or previously unknown secrets.

E Connector HTTP and publishing controls

APX E CLOUD SECRETS

Cloud-stored connector secrets use tenant-specific envelope encryption. This includes supplied API keys, client secrets, private keys, passwords and persisted runtime access tokens. Integration display and editing return secret markers and limited previews, not the full stored secret. Authorized runtime services decrypt values when needed to authenticate to the provider.

HTTP CLIENT + PAGINATION

Cloud connector HTTP destinations require HTTPS. The shared first-party HTTP client follows redirects manually and rejects a change of origin before sending another authenticated request. It also treats pagination links as untrusted: cross-origin links are rejected, page counts are bounded, and stored checkpoints are bound to the tenant, integration and connector version. Administrators approve the initial destination and provider permissions.

PUBLISHING + EGRESS PACING

Custom connectors running in the cloud use the function execution and mediated egress controls described in appendix D. Connector publishing applies schema, rate-policy, conformance and compile checks. Published versions are immutable, and publishing has a permission distinct from authoring. One person may hold both permissions.

The egress coordinator maintains per-tenant, per-integration pacing and provider cooldown state. Provider retry delays are parsed and bounded. The deployed rollout mode determines whether coordination observes traffic or enforces limits.

F Policy gates and provisioning mechanics

APX F
POLICY +
CHANGE
CONTROL

Policy and change control

Deny policies take precedence over grants in the desired-access calculation, including manual grants and access observed through synchronization. When a deny is held for review, existing access is retained, and new grants covered by that deny are not applied while the review is open. Revocation at the external provider is completed separately through provisioning.

Policy authoring and version activation have separate permissions, with one exception: rolling a policy group back disables its member policies under the authoring permission. Managed separation-of-duties constraints generate paired deny policies that ordinary editing paths cannot modify independently. A tenant-wide blast-radius breaker checks projected revocations against a bounded threshold and holds decisions that exceed it for review, including the grants in those decisions.

The default offboarding policy denies access to terminated identities. It is visible and cannot be deleted, although administrators can edit or disable it. Entering the terminated lifecycle state also disables the person's Owlie account in the same database transaction. Removing access from connected systems completes separately.

REQUESTS +
POLICY
AVAILABILITY

Requests, approvals and idempotency

Request preview and submission use the same policy-check logic, including entitlements selected through request forms. A blocking deny requires an appropriately authorized exception; the approval is bound to the specific deny policies presented, so it cannot silently exempt a new deny added later.

When policy evaluation is unavailable, the request engine holds new fulfillment with a participant-visible waiting reason and operator alert. Durable retries reevaluate the complete account, direct entitlement, form-selected entitlement and bundle-component set. If bounded automatic retries exhaust, the request remains held for operator redrive rather than granting access.

The beneficiary cannot approve their own approval ticket or policy exception. Delegated approvers are also blocked when they are the requester or beneficiary. A requester acting for someone else is not automatically prohibited from ordinary approval; the configured approval relationships still matter. Delegation is tenant-scoped and single-hop, with an active human delegate and no self-delegation.

Forms and submissions are validated on the server with bounded sizes and restricted field names. User, group and resource picker references are resolved in the current tenant, regardless of whether the caller submits a bare identifier or a pre-enriched object. Display fields are rebuilt from canonical records; entitlement labels come from the resource catalog. Approval screens use these server-validated records.

Provisioning derives the operation, entitlement set and access window from the canonical request record. Idempotency keys and completed execution-journal entries prevent duplicate submissions and retries from repeating already-recorded successful work. After a crash, ambiguous side effects require reconciliation: operations not known to be safe to repeat are held for manual resolution. If retries are exhausted, the operation stops in an explicit terminal state.

DEFAULT TIMING

APPROVAL WINDOWS

With the default configuration, overdue approvals escalate up the beneficiary's manager chain after `3 days`, approval stages expire `14 days` past due, and a request-wide `90-day` ceiling applies. Tenants and resources can adjust these windows. External providers have their own processing times for fulfillment and revocation.

G Review-event integrity

APX G REVIEWER RESOLUTION

Review subjects cannot certify their own access, including when the subject is an administrator or is reached through a group. Reviewer resolution excludes the subject and uses configured fallback reviewers; an empty fallback fails the item closed. Disabled identities cannot record review decisions. Unanswered reviews remain recorded as no response.

When automation would recreate access after a revoke, the review requires an authorized administrator or operator to record that the source has been addressed before proceeding.

EVIDENCE INTEGRITY

Review evidence has a database-protected event history. A trigger rejects updates to review events, and a partial unique index allows at most one terminal decision per step. Decisions are recorded as events before being projected onto working records.

The governing policy is frozen for an item, and campaign scope, policy and timing are frozen after draft. Working records outside that frozen configuration remain mutable.

H Session, token and connection behavior

APX H
SESSIONS +
MACHINE
CREDENTIALS

Sessions and machine credentials

Session cookies contain an identifier and a random bearer secret. The stored record contains a hash of that secret. Validation checks the tenant and compares the secret hash in constant time. Session cookies use Secure, HttpOnly and SameSite protections.

SESSION LIMITS

Sessions combine a `24-hour` sliding window with a `7-day` absolute limit; a stored upstream SAML session cap can shorten those windows.

API keys provide a separate service-account authentication path. The plaintext key is returned at creation and stored as a SHA-256 digest. Authentication checks the tenant, key status, configured expiry and identity status. Expiry is optional, so administrators must include service-account permissions and key lifecycle in their access-management process.

When Owlie acts as an OpenID Connect provider, it restricts redirect destinations and supports PKCE S256, including required use for public clients. Refresh tokens rotate. The immediate predecessor presented inside the two-minute grace window is rejected without revoking the family; later or non-predecessor reuse revokes the family.

Realtime and browser handling

Realtime connections begin with tenant resolution and authentication and are routed to a tenant-specific object. Session-backed sockets undergo periodic revalidation; an invalid session triggers closure on revalidation, rather than instantaneous logout across all connections. API-key-backed connections require separate revocation handling.

Realtime event payloads are limited to addressing hints such as entity identifiers and coarse statuses, including gateway events. The dashboard retrieves authoritative data through authorized APIs instead of treating an event as permission to read an entity. Events are delivered to an administrator audience or to all users of the tenant, while the API checks permission to read the requested data.

Assistant Markdown rendering disables raw HTML and rejects unsafe URL schemes. Authentication-flow writes use CSRF tokens bound to server-side flow state.

I Audit recording and integrity boundaries

APX I
EVENT
RECORDING

Owlie uses a shared audit service for named product events, including API-key creation and revocation, policy lifecycle actions, exemptions and gateway lifecycle activity. The audit records actor and tenant context so human and service-account activity can be distinguished. For operations that share a database transaction with their audit insert, the operation and audit record commit or fail together.

ACCESS,
RETENTION +
LIMITS

Audit-log access requires a dedicated permission, and queries are scoped to the tenant. Retention is configurable and defaults to `365 days`. A scheduled sweep removes expired records for active and soft-deleted tenants. General audit records do not have cryptographic integrity verification or a database trigger prohibiting modifications by privileged database operators. Retention and authorized tenant erasure are separate from protection against unauthorized changes.

Audit detail keys matching credential-shaped names are redacted before insertion. Oversized detail is replaced by a truncation marker. Key-name matching can miss secrets stored under ordinary field names. The audit viewer uses filtered, paginated queries.

RELATED
TIMELINES

The identity event log separately records attribute and lifecycle changes through explicit emitters. Access-review decisions retain their own event timeline as described in appendix G. Request correlation identifiers connect API responses with the corresponding request logs for investigation.

J Synchronization and notification boundaries

APX J
SYNC
SAFETY

Synchronization safety

Connecting a system does not itself start a sync. Administrators configure the import and its mappings before initiating that work. Cleanup distinguishes records observed from a provider from records Owlie provisioned or people created manually: missing observations can be marked deleted, provisioned records are preserved as missing, and manually created records are not removed by that cleanup.

Filtered or truncated runs are not treated as complete source snapshots. A full run that unexpectedly returns no records after a prior populated run is stopped before cleanup unless an administrator has explicitly disabled that check for the job. Other changes can still produce large deletions. Gateway-staged data also undergoes size, hash and storage-prefix checks before ingestion.

NOTIFICATIONS +
ERROR
HANDLING

Notifications and error handling

Notification delivery resolves identity and group references on the server. Email and Slack rendering escape dynamic content for their respective formats. Inbound Slack actions undergo signature, freshness, body-size and application-identity checks, then reload the referenced entity and authorize the action instead of trusting state supplied by an interactive card.

Rendered content marked sensitive is encrypted through the tenant key service, with failure rather than plaintext fallback when the encryption provider is unavailable. Delivery uses bounded retries, backoff and a dead-letter state. Non-security email that exceeds the tenant's daily budget is deferred rather than immediately discarded.

The GraphQL error boundary replaces unexpected errors with a generic customer-facing message and restricts returned extension fields. The connector SDK sanitizes provider errors, including credential-shaped fields and sensitive URL components. Redaction elsewhere depends on the individual service and log path.

K Backup and recovery mechanics

APX K BACKUP PIPELINE

Owlie's separate backup pipeline encrypts database dumps with an `age` public key inside the backup container before uploading the artifact. The corresponding private key is not required by that backup path. Upload is brokered by a separate Worker; the container does not hold object-storage credentials. The schedule is configured for daily runs, with digests recorded for integrity checking.

RESTORE PROCEDURE

The backup Worker is scheduled daily at `06:00 UTC` and can also be triggered manually. The restore tool downloads an encrypted dump from R2, decrypts it with the matching private key and restores it using PostgreSQL's restore utility. It accepts protected 1Password references for the destination connection and private key.

The private key is held in a restricted vault outside Cloudflare. These dumps cover PostgreSQL data, not a complete copy of all Cloudflare state.

END OF DOCUMENT — SW-01 · VERSION 1.0