API Standards

06/11/2026 4:33 pm GMT

This document defines the baseline standards used for Eyelit's public APIs. The standards are designed to achieve predictable behaviour for integrators, protection of platform resources and customer data, and minimise impacts of upgrades on integration.

Standard API Operations

Each API set exposes the following operations as the default contract. Additional verbs (e.g. bulk operations, search, custom actions) may be added per set, but the five below must exist unless explicitly waived.

Operation	Purpose	HTTP Method
UPSERT	Create or update a resource. Idempotent on the supplied business key.	PUT
LIST	Return a paginated collection, supporting filter, sort, and field selection. Use GET by default, fall back onto POST if criteria length is too long.	GET, POST or QUERY
GET	Return a single resource by identifier.	GET
DELETE	Remove a resource (soft or hard delete to be defined per set).	DELETE
PATCH	Only updates criteria that is sent. Only updates by field.	PATCH

Usage Limits and Protection

All APIs are subject to enforced limits to protect platform stability, customer isolation, and predictable performance under load.

Rate limiting: Every API enforces a request-per-minute ceiling per consumer (API key / customer / agent identity). Limits are configurable per consumer tier. 429 Too Many Requests is returned with a Retry-After header on exceedance.

Payload size: Both byte size and row count are bounded. LIST responses enforce a maximum page size; UPSERT bodies enforce a maximum bytes-per-request and, where applicable, a maximum number of rows per call. The default max payload size is 50MB for standard endpoints, this may be varied by exception. Exceedance returns 413 Payload Too Large.

Timeout: By default a 2 minute server-side timeout will be applied on every request.

Async mode: Long-running processes and queries must be exposed asynchronously. The initial request is validated and accepted with a 202 Accepted response containing an identifier for the request. The caller then uses a separate endpoint to poll for the status and retrieve the result once processing is complete.

Mandatory query constraints: Some APIs enforce required parameters (e.g. a date range, a status filter) to prevent unbounded result sets. These are documented per endpoint and rejected with 400 Bad Request if omitted.

Versioning

We follow semantic versioning (major.minor) at the API level. The rules below define our compatibility commitments to customers.

Contract stability: Minor version updates will never break the contract. Input and response fields are treated differently. For **inputs**: removing a parameter, renaming it, changing its type, changing its required/optional status, or adding a new Mandatory parameter are all major-version changes. For response fields, removing, renaming, or retyping a field is a major-version change.

Minor version increments: Any change to API behaviour or logic — including additive fields, new optional parameters,

performance characteristics, validation rules or response code — increments the minor version.

Major version lifecycle: Each release ships the latest minor version under each supported major version. We continue to release legacy major versions in parallel for as long as they remain compatible with the underlying platform. A deprecation and sunset policy is published with each major release so customers know the supported window.

Default minor version: If a customer does not specify a minor version in the request, the API resolves to the application's configured default minor for the requested major. This ensures customers receive bug fixes and additive improvements automatically without changing integration code.

Customer commitment: Customers should not need to modify integration code or pinned versions unless they want a new major's capabilities or are approaching a sunset deadline.

Version selection: Version is specified via the URL path (e.g. */api/v2/orders*) for the major, with the minor selected via header (X-API-Minor-Version: 3) or query parameter where header-setting is impractical.

Additional Standards

The following standards close gaps that matter particularly for AI-agent consumers, where predictability, machine-readable contracts, and safe retries become first-order concerns.

Authentication and scoping: Every API requires authenticated access.

Idempotency on writes: UPSERT and any non-GET operation that may be retried will accept an Idempotency-Key header. Repeated requests with the same key and body return the original result rather than re-executing.

Consistent error envelope: All error responses follow a single structure including a stable error code, human-readable message, request ID, and where relevant a field-level breakdown.

Pagination contract: LIST endpoints use cursor-based pagination by default (next_cursor, prev_cursor) with a documented default and maximum page size. Offset pagination is permitted only where dataset size and stability guarantee acceptable performance. List functions that are to be paginated will request PAGEINDEX, PAGESIZE and MAXPAGESIZE.

Filtering, sorting, field selection: LIST endpoints support a documented set of filter parameters, a sort parameter, and a fields parameter for sparse responses.

Machine-readable specification: Every API set publishes an OpenAPI 3.x specification.

Stable identifiers and naming: Resource names are plural nouns (/orders, /work-orders). Field naming is CamelCase.

Date and time: All timestamps are ISO 8601 in UTC (2026-05-14T10:15:00Z). If local-time fields are unavoidable, they must be accompanied by an explicit timezone field containing a valid IANA timezone name (e.g. Europe/London, America/New_York).

Request tracing: Every request and response carries a X-Request-Id. Requests inbound from customers may supply one; if absent we generate it. The ID flows through logs and is included in error responses for support diagnostics. String Field Unique ID.

Deprecation signalling: When a field, endpoint, or version is deprecated, responses include a Deprecation header and, where a sunset date is set, a Sunset header (RFC 8594). Deprecations are also documented in release notes a minimum of *[N]* releases before removal.

TLS and transport: TLS 1.3 minimum, HSTS enforced.

Permissions

Every API, without exception, declares and enforces one or more permissions as part of its initial execution path. Failed permission evaluation terminates the request immediately with a Status 403 error (+ message indicating which permission is

needed).

Identity types: The same permission model applies to all calling identities — interactive users, service accounts, integration credentials, and AI agents.

Default deny: Absence of an explicit grant is denial.

Failure responses: Authentication failures return 401 Unauthorized. Authenticated callers who do not have the required permission return 403 Forbidden, using the standard error response envelope and an error code distinct from authentication failures. A consistent 403 response is used for all permission failures, regardless of whether the endpoint targets a collection or a specific resource.

Authenticated users can retrieve a list of all available APIs. Unauthenticated requests return 401 Unauthorized.

Audit trail: Every permission denial is captured in the system log (under security) with caller identity, resource, operation, customer, timestamp, and request ID.

Rate limiting: All endpoints are rate limited per endpoint class. Requests exceeding a limit receive HTTP 429 Too Many Requests with a Retry-After header. Rate limiting is transient and does not change the state of the caller's identity. Repeated permission denials (403) from a single identity raise a security alert and may be throttled, but never trigger lockout.

Credential lockout: Lockout protects against credential guessing and applies only to repeated authentication failures (401), not to permission denials or request volume. After 10 consecutive failed authentication attempts within 15 minutes, the credential is locked for 15 minutes, with the duration increasing on repeat lockouts. Locked credentials receive the same 401 response as invalid credentials, so callers cannot tell whether an account exists or is locked.