VYER.NET // ARCHITECTURE PROPOSAL

EntraProxy Platform API & SDK

A secure, tenant-aware API platform and developer SDK architecture for identity-mediated enterprise integrations.

```
DOCUMENT: EP-API-SDK-001
STATUS: ARCHITECTURE PROPOSAL
CLASSIFICATION: GENERAL / NON-CONFIDENTIAL
```

Executive Brief

EntraProxy requires an API surface that is predictable for integrators, defensible for security teams, and maintainable as product capabilities expand. This proposal defines a versioned ASP.NET Core platform organized around Clean Architecture boundaries, explicit tenant isolation, policy-driven authorization, auditable administrative operations, and generated SDK packages.

The recommended design separates transport, application use cases, domain policy, and infrastructure adapters. External consumers receive stable contracts and consistent errors while internal teams retain the freedom to evolve persistence, identity providers, queues, and telemetry without breaking integrations.

Proposed Architecture

API Layer

ASP.NET Core endpoints, OpenAPI contracts, request validation, version negotiation, rate limits, and standardized problem details.

Application Layer

Commands, queries, authorization decisions, tenant context, orchestration, and transaction boundaries.

Domain Layer

Identity policies, invariants, trust rules, entitlement models, and auditable state transitions.

Infrastructure Layer

Microsoft Entra ID, persistence, secrets, queues, cache, notifications, and observability adapters.

Architecture & Security Findings

01 — Tenant context must be authoritative.
Resolve tenant identity from validated credentials and server-side mappings, never from a client-controlled header alone.

02 — Authorization belongs at the use-case boundary.
Apply resource and action policies beyond route-level role checks to prevent confused-deputy access.

03 — API contracts require explicit versioning.
Adopt additive evolution rules, deprecation windows, and contract tests for supported versions.

04 — Administrative actions need immutable audit events.
Capture actor, tenant, action, target, result, correlation identifier, and trusted timestamp.

05 — SDK generation should follow the canonical OpenAPI document.
Generate typed clients in CI and block releases when generated packages drift from the contract.

06 — Tokens must be audience-bound and short-lived.
Validate issuer, audience, signature, expiry, tenant claims, and required scopes on every request.

07 — Idempotency is required for retryable writes.
Support idempotency keys for creation and workflow endpoints that may be retried by clients or queues.

08 — Errors must not disclose internal topology.
Return stable problem codes and correlation IDs while retaining detailed diagnostics only in protected logs.

09 — Secrets require managed storage and rotation.
Use managed secret stores, workload identities, rotation procedures, and telemetry that never records credentials.

10 — Rate limits must be tenant- and operation-aware.
Protect expensive identity and directory operations separately from inexpensive read endpoints.

11 — Integration boundaries need resilience policies.
Apply bounded retries, timeouts, circuit breaking, and dead-letter handling around Microsoft Graph and downstream systems.

12 — Security verification must be release-gated.
Require dependency scanning, SAST, contract tests, authorization tests, and environment configuration checks in CI/CD.

API Surface

- Tenant and connection lifecycle management
- Identity and directory synchronization
- Policy and entitlement evaluation
- Webhook registration and signed event delivery
- Administrative diagnostics and audit retrieval
- Health, readiness, and version discovery

SDK Packages

EntraProxy.Client

Core generated client, authentication abstractions, retries, pagination, and problem details.

EntraProxy.Extensions.DependencyInjection

ASP.NET Core registration, configuration validation, telemetry, and resilient HTTP defaults.

Roadmap

1. **Foundation:** contract standards, identity model, tenant context, and architecture skeleton.
2. **Core API:** priority use cases, audit events, policy enforcement, and integration adapters.
3. **SDK:** generated packages, hand-authored ergonomics, samples, and package publishing.
4. **Hardening:** performance tests, threat modeling, security gates, and operational runbooks.

Compliance Controls

The design supports least privilege, separation of duties, traceable administrative activity, secure change management, secrets protection, incident investigation, retention controls, and evidence generation relevant to SOC 2 and ISO 27001 programs.

Prepared by Vyer.Net · Security architecture and custom software engineering · wc@vyer.net