

# Project Specification

## Documentation¶

### Constitution¶

This document serves as the supreme law for the project, defining the core values, architectural principles, and decision-making processes that guide all development and specification efforts.

### Vision¶

To create a robust, scalable, and maintainable system that meets the user’s needs with precision and elegance.

### Document Classification¶

This document contains both normative and informative content.

- **Normative** sections define binding rules that **MUST** be followed.
- **Informative** sections provide guidance, context, or rationale.

Unless explicitly stated otherwise, sections under “Specification Conventions” are normative.

### Requirements Language¶

The key words “**MUST**”, “**SHALL**”, “**SHOULD**”, “**MAY**”, and “**MUST NOT**” are to be interpreted as described in RFC 2119 and RFC 8174.

### Core Principles¶

1. **Simplicity**: Favor simple solutions over complex ones. Avoid over-engineering.
2. **Consistency**: Maintain uniformity in code style, naming conventions, and documentation.
3. **Transparency**: All decisions and significant changes must be documented and open for review.

4. **User-Centricity:** The needs of the end-user are paramount in every design decision.

### Architectural Guidelines¶

- **Modularity:** The system should be composed of loosely coupled, highly cohesive modules.
- **Separation of Concerns:** Each component should have a distinct and well-defined responsibility.
- **Scalability:** Design consistently with future growth in mind, but implement for today's requirements.
- **Security First:** Security considerations are integral to the design phase, not an afterthought.

### Decision Making¶

Decisions are made based on technical merit and consensus. When consensus cannot be reached, the project lead has the final authority. All architectural decisions (ADRs) must be recorded.

### Documentation Structure¶

The documentation is organized into the following directories, each serving a specific purpose:

- **doc/spec: Contains the detailed system specifications.**
  - **overview:** High-level summary of the project.
  - **terminology:** Definition of specific terms and abbreviations.
  - **background:** Context, motivations, and problem statement.
  - **scope:** Project boundaries (in-scope and out-of-scope).
  - **actors:** Users and external systems interacting with the project.
  - **use-cases:** Description of functional scenarios and user stories.
  - **functional-requirements:** Specific behaviors and functions the system must support.
  - **non-functional-requirements:** Quality attributes such as performance, security, and reliability.
  - **constraints-and-assumptions:** Limitations and known prerequisites.
  - **data-model:** Entity definitions, database schemas, and data flow.

- **interface-requirements:** UI/UX guidelines and API definitions.
- **error-handling:** Strategies for handling exceptions and failures.
- **future-considerations:** Roadmap and potential future enhancements.
- **doc/adr:** Architecture Decision Records. Used to record significant architectural decisions, context, and consequences.

## Specification Conventions¶

To ensure traceability and maintainability of the specifications, the following conventions SHALL be observed.

**Identifier Scope¶** All identifiers SHALL be globally unique within the project. Identifiers MUST NOT be reused across different categories.

## Identifier Assignment¶

- **Target:** IDs SHALL be assigned to all normative requirements (FR, NFR), use cases (UC), constraints (CON), data entities (DATA), API endpoints (API), and error definitions (ERR).
- **Granularity:** Assign IDs to semantic units, not every paragraph or section.
- **Format:** IDs SHALL follow the format <Category>-<Domain>-<Number>.
  - \* Examples: FR-AUTH-001, NFR-PERF-003, UC-LOGIN-001.
- **Stability:** IDs SHALL NOT change even if the section title or minor wording changes.
- **Deprecation:** If a requirement is removed, keep the ID and mark it as (Deprecated).

**Dependency Model¶** The specification elements are related by the following dependency model.

- Use Cases (UC) provide the business or user justification for Functional Requirements (FR). Every Functional Requirement SHALL be traceable to at least one Use Case.
- Functional Requirements (FR) define the system’s externally observable capabilities and SHALL serve as the central anchor for traceability.
- Error Definitions (ERR) define failure outcomes for Functional Requirements. ERR specifies what happens when an operation fails, not the capability itself. Every Error Definition SHALL be referenced by at least one FR.

- Constraints (CON) define externally imposed, mandatory conditions that restrict or qualify Functional and Non-Functional Requirements. Constraints SHALL NOT be derived from Functional Requirements. Constraints MAY restrict how errors are handled or disclosed.
- Non-Functional Requirements (NFR) define measurable quality attributes of Functional Requirements. Every Non-Functional Requirement SHALL be associated with at least one Functional Requirement. NFRs MAY define retry policies and observability for error conditions.
- Dependencies between UC, FR, ERR, CON, and NFR SHALL form a directed acyclic graph. Cyclic dependencies are prohibited.

#### Traceability Source of Truth¶

- “Impacts” sections SHALL NOT be used. Reverse traceability (e.g., CON → FR/NFR) SHALL be derived by tooling from explicit references.
- Normative traceability SHALL be expressed only by forward references: UC → FR, FR → NFR/CON/ERR, NFR → CON, and (optionally) ERR → CON.
- Constraints (CON) SHALL be declarative and SHALL NOT enumerate impacted elements.

#### Cross-Referencing¶

- **Syntax:** Define targets using .. \_ID: before the header. Reference using :ref:`ID`.
- **Direction:** Follow the reference hierarchy: Use Case -> Functional Requirement -> (API / IF, Data Entity, Error, Constraint). Avoid circular references.
- **Prohibitions:** \* Functional Requirements SHALL NOT reference Use Cases. \* Use Cases SHALL NOT establish normative references to Interface or API specifications (mentions only). \* Data Models SHALL NOT reference API endpoints. \* Error Definitions SHALL NOT introduce new requirements.
- **Semantics:** \* “Realized by” references in Functional Requirements indicate an example implementation mapping and do not constitute a strict normative dependency.

#### Usage Coverage¶

- **FR Coverage:** All Functional Requirements (FR) MUST be referenced by at least one Use Case (UC). Orphan guidelines are considered incomplete specification.

- **UC Completeness:** All Use Cases (UC) MUST reference at least one Functional Requirement (FR). Empty Use Cases are prohibited.
- **Interface Utility:** All Interfaces and APIs MUST be referenced by at least one Functional Requirement (FR). Orphan Interfaces are prohibited.

**Use Case Structure¶** Use Cases (UC) SHALL follow the narrative structure: **Actor -> Entry Point (Informative) -> Goal**. - **Actor:** A defined Actor. - **Entry Point:** A narrative mention of the interface (e.g., “via the Console”). **SHALL NOT** use `:ref:` to link to Interface specifications. - **Goal:** The value or outcome achieved, formally supporting a **Functional Requirement**.

**AI Authoring Rules¶** When generating or modifying documentation:

- New identifiers SHALL NOT be introduced without explicit instruction.
- Existing identifiers SHALL NOT be renamed or repurposed.
- AI-generated content MUST reference existing identifiers where applicable.
- If no suitable identifier exists, the AI SHALL flag the gap instead of inventing one.

**Change Management¶**

- Editorial changes that do not alter meaning SHALL retain the same identifier.
- Semantic changes SHALL result in a new identifier.
- Deprecated identifiers SHALL remain documented and MUST NOT be reused.

**Documentation Format¶**

- **Tables:** Tables SHALL be written using the reStructuredText “Simple Tables” format (using `===` borders) for readability. - Exception: Complex grids that require spanning cells may use “Grid Tables”.

**Amendment Policy¶**

This constitution may be amended as the project evolves. Amendments require a comprehensive review and approval by the core maintainers.

**Specifications¶**

**Overview¶**

**Concept¶** This document outlines the core concept, actors, and scope of the Control Plane System, derived from the initial concept interview.

## Actors¶

- **Tenant User (TERM-ACTOR-USER)**: A user of the customer (tenant) who accesses the SaaS applications. Uses this system for authentication.
- **Platform Operator (TERM-ACTOR-OPS)**: A service provider user who manages subscriptions, configures tenants, and controls feature flags for the SaaS applications.
- **Auditor (TERM-ACTOR-AUDIT)**: A user responsible for reviewing audit logs provided by the system.

## Scope¶ In-Scope

- Issuance of authentication tokens for SaaS applications.
- Delivery of Feature Flags to control application functionality.
- Collection and provision of monitoring and audit logs.
- Recording of billable events/operations.

## Out-of-Scope

- Actual billing and payment processing (e.g., invoice generation, credit card processing via Stripe). These will be handled by an external system based on the recorded events.

**Primary Use Case¶ Tenant Provisioning by Platform Operator** The single most critical scenario is where a **Platform Operator** configures a new tenant and immediately makes the subscribed features available for use. This use case validates the core value of the control plane: rapid and centralized onboarding.

## Terminology¶

**Terminology¶** This section defines key terms and abbreviations used throughout the specification.

**Control Plane (TERM-SYS-CP)¶** The system described in this specification. It provides centralized management functions such as authentication, feature flagging, and auditing for multiple B2B SaaS applications.

**Managed Application (TERM-APP-TARGET)¶** The B2B SaaS applications that are managed by the Control Plane (TERM-SYS-CP). These applications delegate authentication and feature control to the Control Plane. Also referred to as “Target Application” or “B2B SaaS”.

## Background¶¶

**Motivation¶¶** There is a growing need to manage multiple B2B SaaS applications efficiently. Currently, common control plane elements such as User Management (CIAM), Feature Flag management, and monitoring are fragmented or duplicated across applications.

The goal is to centralize these controls to reduce development and operational costs while ensuring consistency across all managed SaaS products.

## Scope¶¶

**Scope¶¶** This section defines the functionalities that are In-Scope and explicitly Out-of-Scope for the project.

### In-Scope¶¶ Authentication (FR-AUTH)¶¶

The system SHALL provide a centralized authentication mechanism (issuance of tokens) for all managed SaaS applications.

### Feature Flag Management (FR-FLAG)¶¶

The system SHALL provide a mechanism to deliver feature flags to applications, enabling dynamic control of functionality.

### Logging and Auditing (FR-LOG)¶¶

The system SHALL collect operational logs and provide them to Auditors. The system SHALL record billable operational events.

### Out-of-Scope¶¶ Payment Processing¶¶

The system SHALL NOT handle handling of actual payments (e.g., credit card transactions) or invoice generation. This is delegated to an external billing system.

### Platform Operator Management¶¶

The management of Platform Operator accounts (registration, deletion, identity management) and their authentication to the Control Plane (IF-OPS-CONSOLE Operator Console) is Out-of-Scope for this specification. These functions are delegated to an external Identity Provider (IdP) and managed by an external team. The system assumes a valid identity is provided via the IdP integration.

## Actors¶¶

**Actors¶¶** This section defines the primary actors interacting with the system.

**ACT-USER Tenant User¶** A user belonging to a tenant organization who accesses the managed B2B SaaS applications. This actor primarily interacts with the authentication services.

**Roles:**

- **Owner:** The primary contact for the tenant. Has full authority, including contract modification (subscription changes) and SSO configuration. Can invite and delete other users.
- **Administrator:** A delegated administrator. Can invite and delete users but cannot modify contracts or configure SSO.
- **User:** A standard user with access to applications but no administrative privileges.

**ACT-OPS Platform Operator¶** An internal user of the service provider responsible for managing the control plane. Responsibilities include tenant onboarding, subscription management, and feature flag configuration.

**ACT-AUDIT Auditor¶** An external or internal compliance officer responsible for reviewing audit logs generated by the system.

**ACT-DEV Developer¶** The engineer or system administrator responsible for deploying and configuring the Managed Application. Interacts with the system to register applications and manage API credentials.

**ACT-BILLING External Billing System¶** A third-party system or internal finance application responsible for processing invoices. Interacts with the system to retrieve billing events.

**Use Cases¶**

**Tenant Provisioning & Lifecycle¶**

**UC-PROV-001 Tenant Provisioning¶** **Actor:** Platform Operator

**Description:** The Platform Operator creates a new tenant configuration and enables subscribed features, allowing the tenant to immediately access the Managed Application.

**Trigger:** A new customer subscription is confirmed.

**Preconditions:**

1. The Platform Operator is logged in.

**Postconditions:**

1. A new tenant entity is created in the Control Plane.



2. Initial Tenant User (Role: Owner) is provisioned.
3. Feature flags corresponding to the subscription plan are active.

**Scenario:**

1. The Platform Operator navigates to the **Operator Console**.
2. The Platform Operator enters tenant details (Name, Domain, Plan) and the email address for the initial Owner.
3. The Platform Operator selects the Managed Application to enable.
4. The Platform Operator selects the “Provision” action.
5. The Control Plane creates the tenant and the initial Tenant User (Role: Owner).
6. The Control Plane enables access to the Managed Application.

**Related Requirements:**

- Flag Configuration
- User Invitation
- Contract Modification
- Control Plane Auditing

**UC-TENANT-SUSPEND Tenant Suspension¶** Actor: Platform Operator

**Description:** The Platform Operator suspends a tenant’s access to the managed application, usually due to non-payment or policy violation.

**Trigger:** The Platform Operator selects “Suspend Tenant” in the **Operator Console**.

**Preconditions:**

1. The Platform Operator is logged in.
2. The target tenant is currently Active.

**Postconditions:**

1. The Tenant status is updated to Suspended.
2. All Users under the tenant are immediately denied access.

**Scenario:**

1. The Platform Operator searches for the tenant in the **Operator Console**.
2. The Platform Operator selects the “Suspend” action.
3. The Platform Operator provides a reason (optional).

4. The Platform Operator confirms the action.
5. The Control Plane invokes the suspension logic.

**Related Requirements:**

- Tenant Status Management
- Control Plane Auditing

**UC-OPS-TENANT-DELETE Tenant Deletion¶** **Actor:** Platform Operator

**Description:** The Platform Operator permanently deletes a tenant and all associated data upon contract termination or deletion request.

**Trigger:** The Platform Operator selects “Delete Tenant” in the **Operator Console**.

**Preconditions:**

1. The Platform Operator is logged in.
2. The target tenant exists.

**Postconditions:**

1. The Tenant enters a 30-day grace period (recoverable).
2. After the grace period, all tenant data is permanently deleted.

**Scenario:**

1. The Platform Operator searches for the tenant in the **Operator Console**.
2. The Platform Operator selects the “Delete” action.
3. The Platform Operator confirms the action with explicit acknowledgment.
4. The Control Plane marks the tenant for deletion (30-day grace).
5. After the grace period, the Control Plane permanently deletes all related data.

**Related Requirements:**

- Tenant Deletion
- Control Plane Auditing
- Data Subject Rights

**Tenant Administration¶**

**UC-TENANT-USER-DELETE User Deletion¶** **Actor:** Tenant User (Role: Owner, Administrator)

**Description:** The Tenant User (Role: Owner or Administrator) removes a user from the tenant organization.

**Trigger:** The Tenant User selects “Delete User” in the **Tenant Administration Console**.

**Preconditions:**

1. The Tenant User is logged in with sufficient privileges.
2. Target user exists.

**Postconditions:**

1. Target user is removed from authentication and cannot access applications.
2. All active Sessions for the user are invalidated.

**Scenario:**

1. The Tenant User selects the user to remove.
2. The Tenant User confirms deletion.
3. The Control Plane removes the user.
4. The Control Plane invalidates existing sessions.

**Related Requirements:**

- User Deletion
- Control Plane Auditing

**UC-TENANT-USER-UPDATE User Role Update¶** **Actor:** Tenant User (Role: Owner, Administrator)

**Description:** The Tenant User (Role: Owner or Administrator) modifies the role of an existing user within the tenant organization.

**Trigger:** The Tenant User selects “Edit Role” in the **Tenant Administration Console**.

**Preconditions:**

1. The Tenant User is logged in with sufficient privileges.
2. Target user exists.

**Postconditions:**

1. Target user’s role is updated.
2. All active Sessions for the user are invalidated.

**Scenario:**

1. The Tenant User selects the user to update.
2. The Tenant User selects the new role.
3. The Tenant User saves the changes.
4. The Control Plane validates the permissions (e.g., cannot downgrade own role if last Owner).
5. The Control Plane updates the user record.
6. The Control Plane invalidates existing sessions to enforce new permissions.

**Related Requirements:**

- User Role Management
- Control Plane Auditing

**UC-TENANT-USER-STATUS-UPDATE User Status Update** ¶ **Actor:** Tenant User (Role: Owner, Administrator)

**Description:** The Tenant User (Role: Owner or Administrator) changes the status of a user (e.g., to Disabled to block access).

**Trigger:** The Tenant User selects “Disable User” or “Enable User” in the **Tenant Administration Console**.

**Preconditions:**

1. The Tenant User is logged in with sufficient privileges.
2. Target user exists.

**Postconditions:**

1. Target user’s status is updated.
2. If Disabled, all active Sessions for the user are invalidated.

**Scenario:**

1. The Tenant User selects the user to update.
2. The Tenant User toggles the status (e.g., Active to Disabled).
3. The Tenant User saves the changes.
4. The Control Plane validates the action.
5. The Control Plane updates the user record.
6. The Control Plane invalidates sessions if required.

**Related Requirements:**

- User Status Management
- Control Plane Auditing

**UC-TENANT-SESSION-REVOKE Session Revocation¶** **Actor:** Tenant User (Role: Owner, Administrator)

**Description:** The Tenant User (Role: Owner or Administrator) invalidates a specific user’s active sessions to force re-authentication.

**Trigger:** The Tenant User selects “Revoke Sessions” for a user in the **Tenant Administration Console**.

**Preconditions:**

1. The Tenant User is logged in with sufficient privileges.
2. The target User exists.

**Postconditions:**

1. All active Sessions for the target User are invalidated.
2. The target User is required to log in again.

**Scenario:**

1. The Tenant User identifies the target user in the **Tenant Administration Console**.
2. The Tenant User initiates the session revocation.
3. The Control Plane invalidates all tokens associated with the user.

**Related Requirements:**

- Session Management
- Control Plane Auditing

**UC-TENANT-INVITE User Invitation¶** **Actor:** Tenant User (Role: Owner, Administrator)

**Description:** The Tenant User (Role: Owner or Administrator) invites a new user to join their tenant organization. The invited user receives an email to set up their account.

**Trigger:** The Tenant User selects “Invite User” in the **Tenant Administration Console**.

**Preconditions:**

1. The Tenant User is logged in with Owner or Administrator role.
2. The invited email address does not already exist in the tenant.

**Postconditions:**

1. An invitation email is sent to the specified address.
2. A user record is created with “Invited” status.

**Scenario:**

1. The Tenant User enters the email address and role (Admin or User) of the new user.
2. The Tenant User submits the invitation.
3. The Control Plane validates the input and permissions.
4. The Control Plane sends the invitation email.

**Related Requirements:**

- User Invitation
- Control Plane Auditing

**UC-TENANT-INVITE-RESEND Invitation Resend¶** **Actor:** Tenant User (Role: Owner or Administrator)

**Description:** The Tenant User resends an invitation to a user whose previous invitation expired or was not received.

**Trigger:** The Tenant User selects “Resend Invitation” for a pending invitation.

**Preconditions:**

1. The Tenant User is logged in with Owner or Administrator role.
2. A pending or expired invitation exists for the target user.

**Postconditions:**

1. The previous invitation is invalidated.
2. A new invitation with fresh expiration is created.
3. A new invitation email is sent to the user.

**Scenario:**

1. The Tenant User views the list of pending invitations.
2. The Tenant User selects “Resend” for the target user.
3. The Control Plane invalidates the old invitation.
4. The Control Plane creates a new invitation and sends the email.

**Related Requirements:**

- Invitation Resend

- Control Plane Auditing

**UC-TENANT-SSO SSO Configuration¶** **Actor:** Tenant User (Role: Owner)

**Description:** The Tenant User (Role: Owner) configures an external Identity Provider (OIDC) to enable Single Sign-On for their users.

**Trigger:** The Tenant User initiates “SSO Setup” in the **Tenant Administration Console**.

**Preconditions:**

1. The Tenant User is logged in with Owner role.
2. The Tenant User has the necessary metadata (Client ID, Issuer URL) from their IdP.

**Postconditions:**

1. The tenant is configured to use the specified IdP.
2. Subsequent logins from this tenant’s domain can use SSO.

**Scenario:**

1. The Tenant User enters IdP details (Issuer URL, Client ID, Client Secret).
2. The Control Plane verifies the IdP configuration (discovery).
3. The Control Plane saves the configuration.

**Related Requirements:**

- Tenant SSO Configuration
- Control Plane Auditing

**Access Management¶**

**UC-LOGIN Tenant User Login¶** **Actor:** Tenant User

**Description:** The Tenant User logs in to the system or a managed application using their credentials or an external IdP.

**Trigger:** The Tenant User attempts to access a protected resource.

**Preconditions:**

1. The Tenant User account exists and is active.

**Postconditions:**

1. The Tenant User receives an authentication token.
2. The Tenant User gains access to the application.

**Scenario:**

1. The Tenant User navigates to the **Universal Login Page**.
2. The Tenant User selects authentication method (Password or SSO).
3. If Password: The Tenant User enters email and password.
4. If SSO: The Tenant User is redirected to IdP and authenticates.
5. The Control Plane validates credentials.
6. The Control Plane issues an authentication token.

**Related Requirements:**

- Supported Authentication Methods
- Session Management

**UC-LOGOUT Tenant User Logout¶ Actor:** Tenant User

**Description:** The Tenant User explicitly terminates their session to secure their access.

**Trigger:** The Tenant User selects “Logout” in the **Universal Login Page** or Managed Application.

**Preconditions:**

1. The Tenant User has an active Session.

**Postconditions:**

1. The Session is invalidated.
2. The user is redirected to the public login page.

**Scenario:**

1. The Tenant User initiates the logout action.
2. The Control Plane invalidates the session token.
3. The Control Plane redirects the user.

**Related Requirements:**

- Session Management
- Control Plane Auditing

**UC-AUTH-RESET Password Reset¶ Actor:** Tenant User

**Description:** The Tenant User initiates a password reset flow when they have forgotten their credentials.



**Trigger:** The Tenant User selects “Forgot Password” on the **Universal Login Page**.

**Preconditions:**

1. The Tenant User has a registered account with an email address.
2. The account is configured for password authentication.

**Postconditions:**

1. The Tenant User has updated their credential.
2. All existing Sessions are invalidated.

**Scenario:**

1. The Tenant User enters their email address on the **Universal Login Page**.
2. The Control Plane sends a password reset link/token to the email.
3. The Tenant User clicks the link and enters a new password.
4. The Control Plane updates the credential store.
5. The Control Plane invalidates all active sessions for the user.

**Related Requirements:**

- Password Reset
- Control Plane Auditing

**Audit Management¶**

**UC-AUDIT-EXPORT Audit Log Export¶** Actor: Auditor

**Description:** The Auditor exports system audit logs for compliance review.

**Trigger:** The Auditor selects “Export Logs” within the **Auditor Console**.

**Preconditions:**

1. The Auditor is logged in with Auditor privileges.

**Postconditions:**

1. A CSV file containing the requested logs is downloaded to the Auditor’s device.

**Scenario:**

1. The Auditor navigates to the “Audit Logs” view in the **Auditor Console**.
2. The Auditor selects the date range and filters for the export.
3. The Auditor initiates the download.

4. The Control Plane queries the log storage.
5. The Control Plane formats the data as CSV and streams the response.

**Related Requirements:**

- Audit Log Collection
- Audit Log Export

**UC-AUDIT-RECORD-CP Control Plane Event Recording¶ Actor:**  
Control Plane

**Description:** The Control Plane records internal state changes (e.g., provisioning, user management) as audit logs to ensure traceability of operator and admin actions.

**Trigger:** A state-changing operation is successfully completed by any Actor.

**Preconditions:**

1. The operation (e.g., Tenant Provisioning, User Deletion) has succeeded.

**Postconditions:**

1. An audit log entry describing the event is persisted.

**Scenario:**

1. An Actor (Operator, Tenant Owner) performs an action (e.g., “Provision Tenant”).
2. The Control Plane executes the detailed business logic.
3. The Control Plane generates an audit log event containing Actor ID, Action, Resource, and Timestamp.
4. The Control Plane persists the log entry.

**Related Requirements:**

- Control Plane Auditing

**System Integration¶** This section describes the automated interactions between Managed Applications and the Control Plane.

**UC-SYS-APP-AUTH Feature Flag Retrieval¶ Actor:** Managed Application (System)

**Description:** The Managed Application retrieves the active feature flags for its tenant to enable/disable functionality dynamically.

**Trigger:** The Managed Application starts up or periodic refresh interval elapses.

**Postconditions:**

1. The Managed Application receives the current set of flags.

**Scenario:**

1. The Managed Application requests flags from the Control Plane.
2. The Control Plane identifies the tenant (via keys or context).
3. The Control Plane returns the flag configuration.

**Related Requirements:**

- Flag Delivery

**UC-SYS-BILL-REPORT Billing Event Reporting¶ Actor:** Managed Application (System)

**Description:** The Managed Application reports a billable event to the Control Plane for tracking.

**Trigger:** A user performs a billable action within the Managed Application.

**Postconditions:**

1. The event is persisted in the Control Plane.

**Scenario:**

1. The Managed Application detects billable event.
2. The Managed Application sends event data to the Control Plane.
3. The Control Plane validates and stores the event.

**Related Requirements:**

- Billing Event Persistence
- Billing Event Ingestion

**UC-SYS-LOG-REPORT Audit Log Reporting¶ Actor:** Managed Application (System)

**Description:** The Managed Application sends its security and operation logs to the Control Plane for centralized auditing.

**Trigger:** The Managed Application generates a log entry.

**Postconditions:**

1. The log entry is collected by the Control Plane.

**Scenario:**

1. The Managed Application generates a log.
2. The Managed Application streams/sends the log to the Control Plane.

3. The Control Plane ingests the log.

**Related Requirements:**

- Audit Log Collection

**UC-SYS-BILL-EXPORT Billing Data Export¶ Actor:** External Billing System (System)

**Description:** The External Billing System retrieves billing events from the Control Plane to generate invoices and perform reconciliation.

**Trigger:** Scheduled job or on-demand request from the External Billing System.

**Postconditions:**

1. The External Billing System receives the requested Billing Events.

**Scenario:**

1. The External Billing System requests billing events for a specified period/tenant.
2. The Control Plane authenticates and authorizes the request.
3. The Control Plane retrieves matching events from the data store.
4. The Control Plane returns the event data to the External Billing System.

**Related Requirements:**

- Billing Data Export

**System Deployment¶**

**UC-DEV-REGISTER Application Registration¶ Actor:** Developer

**Description:** The Developer registers the Managed Application with the Control Plane to obtain credentials for API access. This is typically done as part of the initial system deployment.

**Trigger:** The Developer initiates the “Register System” workflow (via CLI or script).

**Preconditions:**

1. The Developer has administrative access to the Control Plane infrastructure.

**Postconditions:**

1. A Managed Application record is created.
2. An API Access Key is issued and returned to the Developer.
3. The Managed Application is configured with the key.

**Scenario:**

1. The Developer submits the application metadata (Name, Environment).
2. The Control Plane creates the application record.
3. The Control Plane generates a client ID and secret.
4. The Control Plane stores the hashed secret.
5. The Control Plane returns the ID and Secret to the Developer.

**Related Requirements:**

- Application Registration
- API Key Management

**UC-DEV-UPDATE Application Update¶ Actor: Developer**

**Description:** The Developer updates the configuration or status of a Managed Application. This includes disabling the application to prevent access.

**Trigger:** The Developer initiates a deployment or CLI command to update the application.

**Preconditions:**

1. The Developer has valid administrative credentials.
2. The target Managed Application exists.

**Postconditions:**

1. The Managed Application record is updated.
2. If status is set to Disabled, all API access is blocked.

**Scenario:**

1. The Developer submits the update request (e.g., set status to Disabled).
2. The Control Plane validates the request.
3. The Control Plane updates the application record.
4. The Control Plane applies the new state (e.g., blocking incoming API calls).

**Related Requirements:**

- Application Lifecycle Management

**Functional Requirements¶****Authentication & Authorization¶**

**FR-AUTH-001 Supported Authentication Methods¶** The Control Plane (TERM-SYS-CP) SHALL support the following authentication methods for Tenant Users:

- OpenID Connect (OIDC)
- Password-based authentication

**Realized by:**

- Universal Login Page

**Quality Attributes:**

- Encryption in Transit
- MFA
- Least Privilege
- Adaptive Authentication
- Availability SLO

**Error Conditions:**

- Invalid Credentials
- MFA Required
- Rate Limit Exceeded

**FR-AUTH-003 Tenant SSO Configuration¶** The Control Plane (TERM-SYS-CP) SHALL allow a Tenant Owner (role of ACT-USER Tenant User, see Roles) to register an external Identity Provider (IdP) for Single Sign-On (SSO). The configuration SHALL be stored in SSO Configuration.

**Realized by:**

- Tenant Administration Console

**Quality Attributes:**

- Encryption in Transit
- Key Management
- Least Privilege
- Availability SLO

**Error Conditions:**

- Access Denied
- Invalid Input
- Conflict

**FR-AUTH-004 Password Reset¶** The Control Plane (TERM-SYS-CP) SHALL allow Tenant Users (using password authentication) to request a password reset via their registered email address. The Control Plane (TERM-SYS-CP) SHALL allow authenticated Tenant Users to change their password. Upon successful password reset or change, the system SHALL invalidate all active Sessions for the user.

**Realized by:**

- Universal Login Page

**Quality Attributes:**

- Encryption in Transit
- MFA
- Key Management
- Availability SLO

**Error Conditions:**

- Invalid Credentials
- Business Rule Violation
- Rate Limit Exceeded

**FR-AUTH-005 Session Management¶** The Control Plane (TERM-SYS-CP) SHALL establish a Session upon successful user authentication. The Control Plane (TERM-SYS-CP) SHALL support session invalidation triggers including explicit logout and administrative revocation.

**Realized by:**

- Universal Login Page
- Tenant Administration Console

**Quality Attributes:**

- Encryption in Transit
- Encryption at Rest
- Least Privilege
- Availability SLO

**Error Conditions:**

- Access Denied
- Session Expired

**FR-AUTH-006 Password Policy Configuration¶** The Control Plane (TERM-SYS-CP) SHALL allow each Tenant to configure password complexity requirements (e.g., minimum length, required character types).

**Constrained by:** CON-SEC-001 Configurable Password Policy

**Realized by:**

- Tenant Administration Console

**Quality Attributes:**

- Encryption in Transit
- Encryption at Rest
- Least Privilege
- Availability SLO

**Error Conditions:**

- Access Denied
- Invalid Input

**Tenant Administration¶**

**FR-TENANT-001 User Invitation¶** The Control Plane (TERM-SYS-CP) SHALL allow Tenant Owners and Administrators to invite new Users to their Tenant. This process SHALL create a User Invitation record.

- Invitations SHALL have a configurable expiration period (default: 7 days).
- Expired invitations SHALL be marked as invalid and cannot be used for registration.

**Realized by:**

- Tenant Administration Console

**Quality Attributes:**

- Encryption in Transit
- Least Privilege
- User Scalability
- Availability SLO

**Error Conditions:**

- Access Denied
- Conflict



- Business Rule Violation

**FR-TENANT-002 User Deletion¶** The Control Plane (TERM-SYS-CP) SHALL allow Tenant Owners and Administrators to delete Users from their Tenant. When a user is deleted, the system SHALL invalidate all active Sessions for that user.

**Realized by:**

- Tenant Administration Console

**Quality Attributes:**

- Encryption in Transit
- Encryption at Rest
- Least Privilege
- Availability SLO

**Error Conditions:**

- Access Denied
- Resource Not Found

**FR-TENANT-003 Contract Modification¶** The Control Plane (TERM-SYS-CP) SHALL allow only Tenant Owners to modify the tenant's subscription contract (specifically Tenant.plan).

**Realized by:**

- Tenant Administration Console

**Quality Attributes:**

- Encryption in Transit
- Least Privilege
- Tenant Scalability
- Availability SLO

**Error Conditions:**

- Access Denied

**FR-TENANT-004 User Role Management¶** The Control Plane (TERM-SYS-CP) SHALL allow Tenant Owners and Administrators to modify the Roles of existing Users within their Tenant. When a role is updated, the system SHALL invalidate all active Sessions for the target user.

**Realized by:**

- Tenant Administration Console

**Quality Attributes:**

- Encryption in Transit
- Least Privilege
- Availability SLO

**Error Conditions:**

- Access Denied
- Resource Not Found

**FR-TENANT-006 User Status Management¶** The Control Plane (TERM-SYS-CP) SHALL allow Tenant Owners and Administrators to modify the Status of existing Users (e.g., Enable, Disable). When a user is Disabled, the system SHALL invalidate all active Sessions for that user.

**Realized by:**

- Tenant Administration Console

**Quality Attributes:**

- Encryption in Transit
- Least Privilege
- Availability SLO

**Error Conditions:**

- Access Denied
- Resource Not Found

**FR-TENANT-007 Invitation Resend¶** The Control Plane (TERM-SYS-CP) SHALL allow Tenant Owners and Administrators to resend an invitation to a user whose invitation has expired or was not received.

- Resending SHALL invalidate the previous User Invitation and create a new one.
- The new invitation SHALL have a fresh expiration period.

**Realized by:**

- Tenant Administration Console

**Quality Attributes:**

- Encryption in Transit
- Least Privilege

- Availability SLO

**Error Conditions:**

- Access Denied
- Resource Not Found

**Feature Flag Management¶**

**FR-FLAG-001 Flag Configuration¶** The Control Plane (TERM-SYS-CP) SHALL allow ACT-OPS Platform Operator to configure Feature Flags for each Tenant.

**Realized by:**

- Operator Console

**Quality Attributes:**

- Encryption in Transit
- Least Privilege
- Tenant Scalability
- Availability SLO

**Error Conditions:**

- Access Denied
- Resource Not Found

**FR-FLAG-002 Flag Delivery¶** The system SHALL provide an interface via API-FLAG Feature Flag API for Managed Application (TERM-APP-TARGET) to retrieve the current state of Feature Flags.

**Realized by:**

- API-FLAG Feature Flag API

**Quality Attributes:**

- Encryption in Transit
- Key Management
- Availability SLO
- Load Balancing and Failover

**Error Conditions:**

- Invalid Credentials
- Rate Limit Exceeded

## Billing & Usage¶

**FR-BILL-001 Billing Event Persistence¶** The Control Plane (TERM-SYS-CP) SHALL persistently record billable events triggers received via FR-BILL-002 Billing Event Ingestion as Billing Events.

**Realized by:**

- API-BILL Billing Event API

**Quality Attributes:**

- Encryption in Transit
- Encryption at Rest
- Data Residency
- Tenant Scalability
- Availability SLO
- Backup and Redundancy

**Error Conditions:**

- Internal Error
- Service Unavailable

**FR-BILL-002 Billing Event Ingestion¶** The system SHALL provide an API (API-BILL Billing Event API) that allows Managed Application (TERM-APP-TARGET) to report billable events (corresponding to Billing Events).

**Realized by:**

- API-BILL Billing Event API

**Quality Attributes:**

- Encryption in Transit
- Key Management
- Availability SLO
- Load Balancing and Failover

**Error Conditions:**

- Invalid Credentials
- Invalid Input
- Rate Limit Exceeded

**FR-BILL-003 Billing Data Export¶** The Control Plane (TERM-SYS-CP) SHALL provide an interface for External Billing Systems to retrieve Billing Events for invoicing and reconciliation purposes.

**Realized by:**

- API-BILL Billing Event API

**Quality Attributes:**

- Encryption in Transit
- Key Management
- Least Privilege
- Data Residency
- Availability SLO

**Error Conditions:**

- Invalid Credentials
- Access Denied
- Resource Not Found

**Audit & Logging¶**

**FR-LOG-001 Audit Log Collection¶** The Control Plane (TERM-SYS-CP) SHALL collect security and operational logs from all components via API-LOG Audit Log API and persist them as Audit Logs.

**Realized by:**

- API-LOG Audit Log API

**Quality Attributes:**

- Encryption in Transit
- Encryption at Rest
- Data Residency
- Availability SLO
- Backup and Redundancy
- Continuous Monitoring

**Error Conditions:**

- Internal Error
- Service Unavailable

**FR-LOG-002 Audit Log Export¶** The Control Plane (TERM-SYS-CP) SHALL allow ACT-AUDIT Auditor to export Audit Logs in CSV format.

**Realized by:**

- Auditor Console

**Quality Attributes:**

- Encryption in Transit
- Least Privilege
- Data Residency
- Availability SLO

**Error Conditions:**

- Access Denied
- Internal Error

**FR-LOG-003 Control Plane Auditing¶** The Control Plane (TERM-SYS-CP) SHALL record its own state-changing operations (e.g., Tenant Provisioning, User Management) as Audit Logs.

**Quality Attributes:**

- Encryption at Rest
- Data Residency
- Availability SLO
- Continuous Monitoring

**Error Conditions:**

- Internal Error

**Platform Operations¶**

**FR-OPS-001 Tenant Status Management¶** The Control Plane (TERM-SYS-CP) SHALL allow ACT-OPS Platform Operator to modify the status of a Tenant (e.g., Active, Suspended). When a Tenant is Suspended, the system SHALL revoke access for all Users associated with that tenant.

**Realized by:**

- Operator Console

**Quality Attributes:**

- Encryption in Transit

- Least Privilege
- Tenant Scalability
- Availability SLO

**Error Conditions:**

- Access Denied
- Resource Not Found

**FR-OPS-002 Tenant Deletion¶** The Control Plane (TERM-SYS-CP) SHALL allow ACT-OPS Platform Operator to permanently delete a Tenant and all associated data upon contract termination or deletion request.

- Upon deletion request, the tenant SHALL enter a 30-day grace period (recoverable).
- After the grace period, all tenant data SHALL be permanently and irrecoverably deleted.
- Deleted data includes: Users, Sessions, Audit Logs, Billing Events, and configuration.

**Constrained by:** CON-COMP-001 Data Subject Rights

**Realized by:**

- Operator Console

**Quality Attributes:**

- Encryption in Transit
- Least Privilege
- Availability SLO

**Error Conditions:**

- Access Denied
- Resource Not Found
- Resource Gone

**FR-OPS-003 Data Retention Enforcement¶** The Control Plane (TERM-SYS-CP) SHALL automatically enforce data retention policies:

- Audit Logs SHALL be retained for a minimum of 7 years, then securely deleted.
- Billing Events SHALL be retained for a minimum of 5 years, then securely deleted.

**Constrained by:** CON-DATA-002 Audit Log Retention, CON-DATA-003 Billing Data Retention

**Realized by:**

- Control Plane (TERM-SYS-CP) (automated process)

**Quality Attributes:**

- Encryption at Rest
- Data Residency
- Availability SLO

**Error Conditions:**

- Internal Error

## **System Operations¶**

**FR-SYS-001 Application Registration¶** The Control Plane (TERM-SYS-CP) SHALL allow ACT-DEV Developer to register a new Managed Application. The system SHALL generate a unique Application ID upon registration.

**Quality Attributes:**

- Encryption in Transit
- Key Management
- Least Privilege
- Availability SLO

**Error Conditions:**

- Access Denied
- Invalid Input
- Conflict

**FR-SYS-002 API Key Management¶** The Control Plane (TERM-SYS-CP) SHALL allow ACT-DEV Developer to issue API Access Keys for a registered Managed Application. The system SHALL display the Client Secret only once upon issuance. The system SHALL allow ACT-DEV Developer to revoke existing keys.

**Quality Attributes:**

- Encryption in Transit
- Encryption at Rest
- Key Management



- Least Privilege
- Availability SLO

**Error Conditions:**

- Access Denied
- Resource Not Found

**FR-SYS-003 Application Lifecycle Management¶** The Control Plane (TERM-SYS-CP) SHALL allow ACT-DEV Developer to update the configuration of a Managed Application. The Control Plane (TERM-SYS-CP) SHALL allow ACT-DEV Developer to change the Status (e.g., Disable) to block access.

**Quality Attributes:**

- Encryption in Transit
- Least Privilege
- Availability SLO

**Error Conditions:**

- Access Denied
- Resource Not Found

**FR-SYS-004 Operator JIT Provisioning¶** The Control Plane (TERM-SYS-CP) SHALL automatically provision a User record for Platform Operators upon their first successful authentication via the external Identity Provider.

- The provisioned user SHALL have `tenant_id` set to NULL (platform-level user).
- The user's `id` SHALL be used as `actor_id` in Audit Logs.
- Subsequent logins SHALL reuse the existing user record.

**Quality Attributes:**

- Encryption in Transit
- Least Privilege
- Availability SLO
- Continuous Monitoring

**Error Conditions:**

- Invalid Credentials
- Internal Error

## Non-Functional Requirements¶

This section defines quality attributes that apply across the system.

## Security¶

**NFR-SEC-001 Encryption in Transit¶** All network communications SHALL use TLS 1.2 or higher.

**Constrained by:** CON-SEC-003 Encryption in Transit

**NFR-SEC-002 Encryption at Rest¶** All persistent data SHALL be encrypted using AES-256 or equivalent algorithm.

**Constrained by:** CON-SEC-004 Encryption at Rest

**NFR-SEC-003 Multi-Factor Authentication¶** The system SHALL support Multi-Factor Authentication (MFA) for user login. MFA options SHALL include TOTP (Time-based One-Time Password) and WebAuthn.

**Constrained by:** CON-SEC-005 Multi-Factor Authentication

**NFR-SEC-004 Key Management¶** The system SHALL implement secure key management practices including:

- Automated key generation using cryptographically secure methods.
- Periodic key rotation (at least annually).
- Secure key destruction upon expiration or revocation.

**Constrained by:** CON-SEC-006 Key Management

**NFR-SEC-005 Least Privilege¶** The system SHALL enforce the principle of least privilege:

- Users and services SHALL be granted only the minimum permissions required.
- Separation of duties SHALL be implemented for sensitive operations.

**Constrained by:** CON-SEC-007 Least Privilege

**NFR-SEC-006 Adaptive Authentication¶** The system SHOULD implement adaptive (risk-based) authentication that considers:

- User location and device.
- Time of access.
- Behavioral patterns.

When elevated risk is detected, additional authentication factors SHALL be required.

**Constrained by:** CON-SEC-008 Adaptive Authentication

## **Data¶¶**

**NFR-DATA-001 Data Residency¶¶** Customer data SHALL be stored exclusively in data centers located in Japan or the European Union.

**Constrained by:** CON-DATA-001 Data Residency

## **Capacity¶¶**

**NFR-CAP-001 Tenant Scalability¶¶** The system SHALL scale to support a minimum of 1,000 concurrent active Tenants.

**Constrained by:** CON-CAP-001 Tenant Capacity

**NFR-CAP-002 User Scalability¶¶** The system SHALL scale to support a minimum of 10,000 Users per Tenant.

**Constrained by:** CON-CAP-002 User Capacity per Tenant

## **Availability¶¶**

**NFR-OPS-001 Service Level Objective¶¶** The system SHALL maintain 99.9% availability, excluding scheduled maintenance windows.

**Constrained by:** CON-OPS-001 Availability SLO

**NFR-OPS-002 Maintenance Scheduling¶¶** Scheduled maintenance windows SHALL be defined and communicated to stakeholders in advance. (Details TBD)

**Constrained by:** CON-OPS-002 Maintenance Window

**NFR-OPS-003 Load Balancing and Failover¶¶** The system SHALL implement:

- Load balancing to distribute traffic across multiple instances.
- Automatic failover to prevent single points of failure.
- Geographic redundancy where feasible.

**Constrained by:** CON-OPS-003 Load Balancing and Failover

**NFR-OPS-004 Backup and Redundancy¶** The system SHALL implement backup and recovery capabilities:

- Automated regular backups (at least daily).
- 3-2-1 backup rule (3 copies, 2 different media, 1 offsite).
- Backup encryption.
- Regular restore testing (at least annually).

**Constrained by:** CON-OPS-004 Backup and Redundancy

## **Performance¶**

**NFR-PERF-001 Authentication Latency¶** Authentication requests SHALL complete within the following latency targets:

- P95 (95th percentile): < 1000ms
- P99 (99th percentile): < 2000ms

This includes password verification, OIDC/SSO flows, and MFA validation.

**Constrained by:** CON-PERF-001 Response Time and Throughput

**NFR-PERF-002 API Latency¶** General API requests SHALL complete within the following latency targets:

- P95 (95th percentile): < 500ms
- P99 (99th percentile): < 1000ms

**Constrained by:** CON-PERF-001 Response Time and Throughput

**NFR-PERF-003 UI Responsiveness¶** User interface interactions SHALL provide feedback within the following targets:

- Page transitions: < 2 seconds
- Interactive elements: < 400ms perceived response

**Constrained by:** CON-PERF-001 Response Time and Throughput

**NFR-PERF-004 Authentication Throughput¶** The authentication system SHALL support a minimum throughput of:

- 5,000 authentication requests per second (RPS)

This ensures capacity for peak login periods across all tenants.

**Constrained by:** CON-PERF-001 Response Time and Throughput

## **Monitoring¶**

**NFR-MON-001 Continuous Monitoring¶** The system SHALL implement continuous monitoring capabilities:

- Real-time network traffic analysis.
- User and Entity Behavior Analytics (UEBA) for anomaly detection.
- Automated alerting for suspicious activities.

**Constrained by:** CON-MON-001 Continuous Monitoring

### **Constraints and Assumptions¶**

This section documents the mandatory conditions imposed on the system.

### **Security Constraints¶**

**CON-SEC-001 Configurable Password Policy¶** The Control Plane (TERM-SYS-CP) SHALL allow each Tenant to configure password complexity requirements (e.g., minimum length, character types).

**Rationale:** Compliance with Japan's Act on the Protection of Personal Information (APPI).

**CON-SEC-002 Audit Log Provision¶** The Control Plane (TERM-SYS-CP) SHALL provide audit logs recording access to and modifications of personal data.

**Rationale:** Compliance with Japan's Act on the Protection of Personal Information (APPI).

**CON-SEC-003 Encryption in Transit¶** All communications between components and external systems SHALL be encrypted using TLS 1.2 or higher.

**Rationale:** NIST CSF 2.0 / ISO 27001 - Encryption in transit.

**CON-SEC-004 Encryption at Rest¶** All persistent data SHALL be encrypted at rest using AES-256 or equivalent.

**Rationale:** NIST CSF 2.0 / ISO 27001 - Encryption at rest.

**CON-SEC-005 Multi-Factor Authentication¶** The system SHALL support Multi-Factor Authentication (MFA) for user login.

**Rationale:** NIST CSF 2.0 - Strong authentication.

**CON-SEC-006 Key Management¶** The system SHALL implement secure key management practices.

**Rationale:** NIST CSF 2.0 - Data protection.

**CON-SEC-007 Least Privilege¶** The system SHALL enforce the principle of least privilege.

**Rationale:** NIST CSF 2.0 - Access control.

**CON-SEC-008 Adaptive Authentication¶** The system SHOULD implement adaptive (risk-based) authentication.

**Rationale:** NIST CSF 2.0 - Context-aware security.

**CON-SEC-009 Error Disclosure¶** Error responses SHALL NOT reveal sensitive internal details including:

- Stack traces or internal exception messages
- Database query details
- Internal service names or versions
- User existence confirmation (for security-sensitive operations)

**Rationale:** NIST CSF 2.0 / OWASP - Information disclosure prevention.

#### **Data Constraints¶**

**CON-DATA-001 Data Residency¶** All customer data SHALL be stored exclusively in data centers located in Japan or the European Union.

**Rationale:** GDPR / APPI - Cross-border data transfer restrictions.

**CON-DATA-002 Audit Log Retention¶** Audit Logs SHALL be retained for a minimum of 7 years to comply with Japanese accounting laws and APPI requirements.

**Rationale:** Legal compliance (Japan Accounting Law, APPI).

**CON-DATA-003 Billing Data Retention¶** Billing Events SHALL be retained for a minimum of 5 years to comply with tax regulations.

**Rationale:** Legal compliance (Tax regulations).

#### **Capacity Constraints¶**

**CON-CAP-001 Tenant Capacity¶** The system SHALL support a minimum of 1,000 active Tenants.

**Rationale:** Business scalability requirement.

**CON-CAP-002 User Capacity per Tenant¶** The system SHALL support a minimum of 10,000 Users per Tenant.

**Rationale:** Business scalability requirement.

#### **Operational Constraints¶**

**CON-OPS-001 Availability SLO¶** The system SHALL target a Service Level Objective (SLO) of 99.9% availability, excluding scheduled maintenance windows.

**Rationale:** Business continuity requirement.

**CON-OPS-002 Maintenance Window¶** Scheduled maintenance windows SHALL be defined and communicated in advance. (Details TBD)

**Rationale:** Operational planning.

**CON-OPS-003 Load Balancing and Failover¶** The system SHALL implement load balancing and automatic failover.

**Rationale:** NIST CSF 2.0 - Availability.

**CON-OPS-004 Backup and Redundancy¶** The system SHALL implement backup and recovery capabilities following the 3-2-1 rule.

**Rationale:** NIST CSF 2.0 - Recovery.

#### **Monitoring Constraints¶**

**CON-MON-001 Continuous Monitoring¶** The system SHALL implement continuous monitoring and anomaly detection.

**Rationale:** NIST CSF 2.0 - Detection.

#### **Development Constraints¶**

**CON-DEV-001 Secure Software Development Lifecycle¶** All software development SHALL follow Secure SDLC practices:

- Security by Design principles.
- Static and dynamic code analysis.
- Code review for security vulnerabilities.
- Dependency vulnerability scanning.

**Rationale:** NIST CSF 2.0 - Software integrity.

**CON-DEV-002 Supply Chain Risk Management¶** The project SHALL implement Cyber Supply Chain Risk Management (C-SCRM):

- Inventory of third-party dependencies.
- Regular vulnerability assessment of dependencies.
- Risk evaluation of third-party service providers.

**Rationale:** NIST CSF 2.0 - Supply chain security.

### Compliance Constraints¶

**CON-COMP-001 Data Subject Rights¶** The system SHALL support data subject rights as required by GDPR and Japan APPI:

- Right to access personal data.
- Right to rectification.
- Right to erasure (right to be forgotten).
- Data portability.

**Rationale:** Legal compliance with privacy regulations.

### Performance Constraints¶

**CON-PERF-001 Response Time and Throughput¶** The system SHALL meet performance targets for latency and throughput to ensure a responsive user experience.

**Rationale:** User-Centricity principle - responsive systems improve user satisfaction.

### Data Model¶

**Schema Definitions¶** This section defines the core data entities managed by the Control Plane.

**Tenants¶** Represents a customer organization subscribed to the Managed Application.

Field	Type	Description
id	UUID	Unique identifier for the tenant.
name	String	Display name of the organization.
domain	String	Unique domain identifier for the tenant (e.g., <b>acme</b> ).
plan	Enum	Subscription plan (e.g., <b>Free</b> , <b>Pro</b> , <b>Enterprise</b> ).
status	Enum	Account status ( <b>Active</b> , <b>Suspended</b> ).



Field	Type	Description
created_at	Timestamp	Record creation time.

**Users** ¶ Represents an individual user belonging to a Tenant or the Platform.

Field	Type	Description
id	UUID	Unique identifier for the user.
tenant_id	UUID	Foreign Key to Tenants. Null for Platform Operators.
email	String	Unique email address used for login.
role	Enum	Access level (Roles).
status	Enum	User status ( <b>Invited</b> , <b>Active</b> , <b>Disabled</b> ).

**Roles** ¶ Enumeration of defined user roles.

- **Owner**: Full access to tenant configuration, billing, and user management.
- **Administrator**: Access to user management and tenant configuration (excluding billing).
- **User**: Access to the Managed Application features only.
- **Operator**: (Platform level) Full access to the Control Plane.

**Feature Flags** ¶ Controls the availability of features for specific tenants.

Field	Type	Description
id	UUID	Unique identifier.
tenant_id	UUID	Foreign Key to Tenants.
key	String	Feature identifier (e.g., <b>ai_module_enabled</b> ).
value	Boolean	State of the feature (True/False).

**Audit Logs** ¶ Immutable record of system events for security and compliance.

Field	Type	Description
id	UUID	Unique identifier.
timestamp	Timestamp	Time when the event occurred.
actor_id	String	ID of the user or system component initiating the action.
actor_type	Enum	Type of actor ( <b>User</b> , <b>Operator</b> , <b>System</b> ).
action	String	Description of the operation (e.g., <b>tenant.create</b> ).
resource	String	Identifier of the target resource.

Field	Type	Description
outcome	Enum	Result of the operation ( <b>Success</b> , <b>Failure</b> ).
metadata	JSON	Additional context (e.g., previous values, IP address).

**Managed Applications**¶ Represents a registered Managed Application instance that interacts with the Platform APIs.

Field	Type	Description
id	UUID	Unique identifier.
name	String	Name of the application.
owner_id	String	Identifier of the Developer or owner.
status	Enum	Registration status ( <b>Active</b> , <b>Disabled</b> ).

**API Access Keys**¶ Credentials used by Managed Applications to authenticate against Control Plane APIs.

Field	Type	Description
id	UUID	Unique key identifier (KID).
app_id	UUID	Foreign Key to Managed Applications.
key_hash	String	Secure hash of the API Secret.
scopes	String[]	List of allowed API scopes (e.g., <b>bill:write</b> , <b>log:write</b> ).
created_at	Timestamp	Issuance time.
expires_at	Timestamp	Expiration time (optional).

**SSO Configuration**¶ Stores the Identity Provider details for a Tenant.

Field	Type	Description
id	UUID	Unique identifier.
tenant_id	UUID	Foreign Key to Tenants.
issuer_url	String	OIDC Issuer URL.
client_id	String	Client Identifier at IdP.
secret	String	Encrypted Client Secret.
created_at	Timestamp	Configuration time.

**User Invitations**¶ Tracks pending invitations for new users.

Field	Type	Description
id	UUID	Unique identifier.
tenant_id	UUID	Foreign Key to Tenants.
email	String	Target email address.
role	Enum	Proposed role (Roles).
token	String	Unique token sent via email.
expires_at	Timestamp	Token expiration time.
status	Enum	Invitation status ( <b>Pending</b> , <b>Accepted</b> , <b>Expired</b> ).

**Billing Events**¶ Raw records of billable activities reported by applications.

Field	Type	Description
id	UUID	Unique identifier.
tenant_id	UUID	Foreign Key to Tenants.
app_id	UUID	Foreign Key to Managed Applications.
event_type	String	Type of billable action (e.g., <b>api_call</b> , <b>storage_gb</b> ).
quantity	Integer	Amount consumed.
timestamp	Timestamp	Event occurrence time.

**User Sessions**¶ Represents an active login session for a User.

Field	Type	Description
id	UUID	Unique identifier for the session.
user_id	UUID	Foreign Key to Users.
token	String	Secure session token or JWT reference.
created_at	Timestamp	Session creation time.
expires_at	Timestamp	Session expiration time.
status	Enum	Session status ( <b>Active</b> , <b>Revoked</b> ).

## Interface Requirements¶

**User Interfaces**¶ This section defines the primary User Interfaces (UI) provided by the Control Plane.

**IF-OPS-CONSOLE Operator Console**¶ **User:** ACT-OPS Platform Operator **Description:** The administrative web portal for Platform Operators. Provides capabilities for tenant provisioning, feature flag management, and system monitoring.

**IF-TENANT-CONSOLE Tenant Administration Console¶** **User:** ACT-USER Tenant User (Owner, Admin) **Description:** The self-service web portal for Tenant Administrators. Provides capabilities to list users, manage roles and status (e.g., Disable), revoke sessions, send invitations, configure SSO, and view subscription details.

**IF-AUDIT-CONSOLE Auditor Console¶** **User:** ACT-AUDIT Auditor **Description:** The compliance and observation portal for Auditors. Provides read-only access to system audit logs and reporting capabilities.

**IF-LOGIN-UI Universal Login Page¶** **User:** ACT-USER Tenant User, ACT-OPS Platform Operator, ACT-AUDIT Auditor **Description:** The centralized login page presented to all users. Supports input for Email/Password, redirects for SSO/OIDC authentication, and provides access to password reset workflows.

**Interface Requirements¶** This section defines the external interfaces provided by the system.

**API-BILL Billing Event API¶** **Type:** REST API **Direction:** Bidirectional (Managed Application (TERM-APP-TARGET) <-> Control Plane (TERM-SYS-CP) / ACT-BILLING External Billing System <- Control Plane (TERM-SYS-CP)) **Purpose:** To report billable operations from managed applications and to retrieve billing events for invoicing. **Payload:** SHALL include Tenant ID, Timestamp, Event Type, and Quantity.

**API-LOG Audit Log API¶** **Type:** REST API **Direction:** Input (Managed Application (TERM-APP-TARGET) -> Control Plane (TERM-SYS-CP)) **Purpose:** To report security and operational events for audit purposes. **Payload:** SHALL include Timestamp, Actor ID, Event Type, Resource ID, Outcome, and IP Address.

**API-FLAG Feature Flag API¶** **Type:** REST API **Direction:** Output (Control Plane (TERM-SYS-CP) -> Managed Application (TERM-APP-TARGET)) **Purpose:** To retrieve the active feature flags for a specific tenant. **Caching:** Managed apps SHOULD cache this response to minimize latency.

## Error Handling¶

This section documents error definitions. Errors define failure outcomes for Functional Requirements - what happens when an operation fails.

**Error Definition Structure¶** Each error SHALL include:

- **Description:** Semantic meaning of the error

- **Triggers:** Conditions that cause the error
- **Outcome:** Observable result when the error occurs
- **Related Requirements:** FRs that may produce this error
- **Constraints:** CONs that restrict error handling (if applicable)

#### **Authentication & Authorization Errors¶** ERR-AUTH-401 Invalid Credentials¶

**Description:** Authentication failed due to invalid credentials.

**Triggers:**

- Incorrect password
- Invalid or expired authentication token
- Unknown username/email

**Outcome:**

- Authentication is denied
- No session is established

**Constraints:**

- Error Disclosure

#### **ERR-AUTH-403 Access Denied¶**

**Description:** An authenticated user attempted an operation they are not authorized to perform.

**Triggers:**

- Insufficient role or permissions
- Attempting to access another tenant's resources
- Disabled user account

**Outcome:**

- Operation is rejected
- No state change occurs

**Constraints:**

- Error Disclosure

#### **ERR-AUTH-440 Session Expired¶**

**Description:** The user's session has expired or been invalidated.

**Triggers:**

- Session timeout
- Administrative session revocation
- Password change
- User deletion or disabling

**Outcome:**

- Current operation is rejected
- User must re-authenticate

**ERR-AUTH-461 MFA Required**

**Description:** Multi-factor authentication is required but not provided.

**Triggers:**

- MFA is enabled for the user/tenant
- Adaptive authentication detected elevated risk

**Outcome:**

- Authentication flow pauses
- User is prompted for MFA verification

**Constraints:**

- MFA

**Validation Errors** ERR-VAL-400 Invalid Input

**Description:** The request contains invalid or malformed input data.

**Triggers:**

- Missing required fields
- Invalid data format
- Value out of allowed range

**Outcome:**

- Operation is rejected
- No state change occurs

**ERR-VAL-409 Conflict**

**Description:** The operation conflicts with existing state.

**Triggers:**

- Duplicate email address in invitation

- Attempting to create a resource that already exists
- Concurrent modification conflict

**Outcome:**

- Operation is rejected
- Existing state is preserved

**ERR-VAL-422 Business Rule Violation¶**

**Description:** The operation violates a business rule or policy.

**Triggers:**

- Password does not meet complexity requirements
- Invitation has expired
- Tenant capacity exceeded

**Outcome:**

- Operation is rejected
- No state change occurs

**Resource Errors¶ ERR-RES-404 Resource Not Found¶**

**Description:** The requested resource does not exist.

**Triggers:**

- Invalid resource ID
- Resource has been deleted
- Resource belongs to a different tenant

**Outcome:**

- Operation is rejected
- No state change occurs

**Constraints:**

- Error Disclosure

**ERR-RES-410 Resource Gone¶**

**Description:** The resource previously existed but has been permanently deleted.

**Triggers:**

- Tenant has been deleted (past grace period)
- Data retention policy applied

**Outcome:**

- Operation is rejected
- Resource cannot be recovered

**Rate Limit Errors¶ ERR-RATE-429 Rate Limit Exceeded¶**

**Description:** The client has exceeded the allowed request rate.

**Triggers:**

- Too many requests in a short period
- Authentication attempt rate limiting
- API call quota exceeded

**Outcome:**

- Current request is rejected
- Client should wait and retry

**Constraints:**

- Response Time and Throughput

**System Errors¶ ERR-SYS-500 Internal Error¶**

**Description:** An unexpected internal error occurred.

**Triggers:**

- Unhandled exception
- Database connection failure
- Dependency service unavailable

**Outcome:**

- Operation fails
- State may be inconsistent (should be recoverable)

**Constraints:**

- Error Disclosure

**ERR-SYS-503 Service Unavailable¶**

**Description:** The service is temporarily unavailable.

**Triggers:**

- Scheduled maintenance
- System overload



- Failover in progress

**Outcome:**

- Operation is rejected
- Client should retry later

**Constraints:**

- Maintenance Window

**Future Considerations¶**

**Roadmap¶** This section outlines potential future enhancements and features that are currently out of scope but may be considered for future releases.

**FC-AUTH-SAML SAML Authentication Support¶** **Status:** Future Consideration **Description:** Support for Security Assertion Markup Language (SAML) 2.0 based Single Sign-On (SSO). Currently, the system only supports OIDC for simplicity and modern standards compliance, but SAML remains a common requirement for enterprise customers.

**Architecture Description¶**

**Purpose** This document describes the architecture of the system, focusing on *how* it satisfies the requirements defined in the Specification.

**Stakeholders** - Developers - Architects - System Operators

**Viewpoints** This documentation follows the **arc42** template and utilizes **C4 model** diagrams.

Contents:

**Introduction and Goals¶**

This section outlines the driving forces behind the architecture.

**Requirements Overview¶** The system's functional and non-functional requirements are detailed in the Specification.

See Background and Motivation for the project context.

**Quality Goals¶** The architecture prioritizes the following NFR categories:

- Security
- Performance
- Availability

Refer to the specific NFR documents for detailed metrics.

## Architecture Constraints¶

This section references the constraints that limit the architectural freedom.

**No New Constraints** This architecture documentation introduces **no new constraints**. All binding constraints are defined in the Specification.

**Normative Constraints¶** Refer to Constraints and Assumptions for the authoritative definitions.

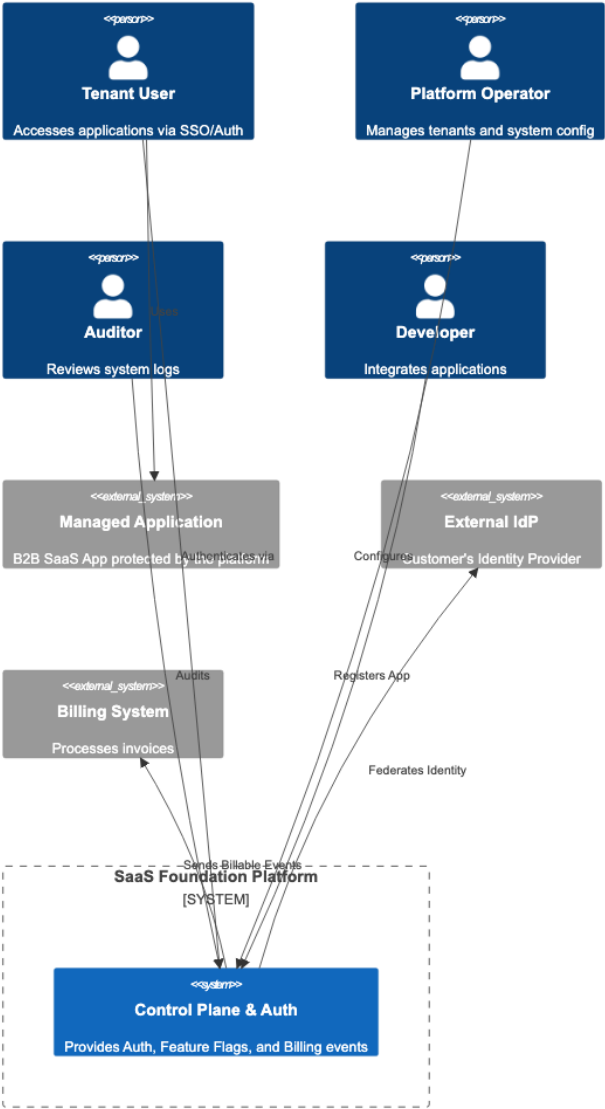
- **Constraints:** CON-\* declarations.
- **Assumptions:** ASM-\* declarations.

## Context View¶

This section describes the system's boundaries and interactions with external actors and systems.



System Context Diagram for SaaS Foundation Platform



System Context Diagram (level 1)¶

**External Elements¶** The following actors and systems interact with the SaaS Foundation Platform.

**Actors¶**

- **Tenant User:** ACT-USER
- **Platform Operator:** ACT-OPS
- **Auditor:** ACT-AUDIT
- **Developer:** ACT-DEV

**External Systems¶**

- **Managed Application:** TERM-APP-TARGET
- **External Billing System:** ACT-BILLING

**Scope Boundaries¶**

- **In-Scope:** Authentication, Feature Flags, Logging (See Scope).
- **Out-of-Scope:** Payment Processing, Operator IdP Management.

**Solution Strategy¶**

This section describes the fundamental decisions and solution strategies that shape the architecture.

**Architectural Patterns¶** The system follows a **Modular Monolith** or **Service-Oriented Architecture (SOA)** approach, centered around a centralized **Control Plane**.

- **Separation of Concerns:** The Control Plane is strictly separated from the Managed Applications.
- **API-First:** All functionality is exposed via defined APIs (API-\*).

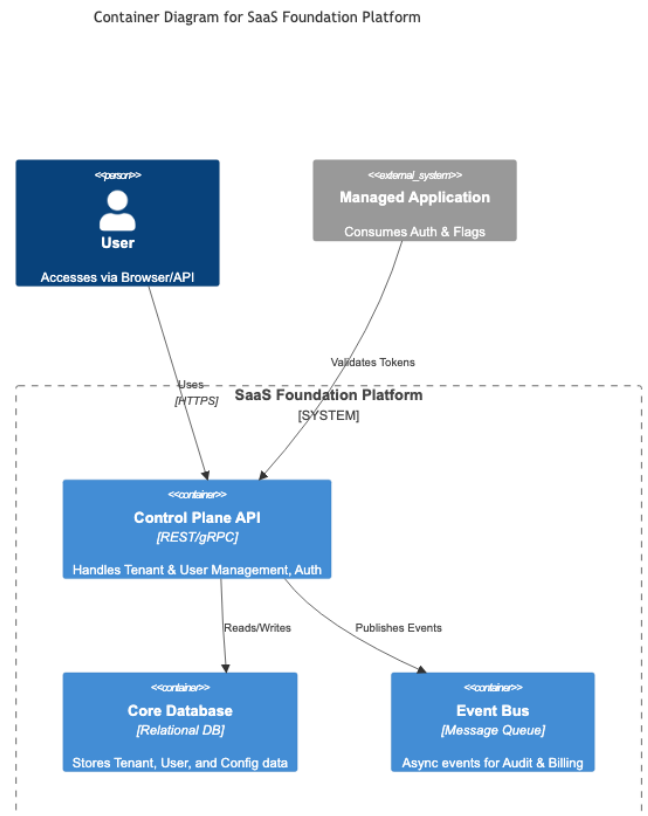
**Quality Goals Strategy¶** This strategy addresses the key NFRs as follows:

- **Security:** Centralized Authentication (FR-AUTH) ensures consistent identity management across all apps. See NFR-SEC-\*.
- **Auditability:** Sync/Async event recording (FR-LOG) ensures all critical actions are traceable.
- **Extensibility:** Feature Flags (FR-FLAG) allow dynamic feature management without redeployment.

**Technology Independence¶** This architecture does not mandate specific programming languages or frameworks, provided they satisfy the Interface Requirements.

## Building Block View¶

This section decomposes the system into its building blocks (modules, components).



## Level 1: System Whitebox¶

### Container Responsibilities:

1. **Control Plane API - Responsibilities:** Authentication, Tenant Management, Feature Flag distribution. - **Satisfies:** FR-AUTH, FR-FLAG.
2. **Core Database - Responsibilities:** Persistence of normative data models.

- **Satisfies:** Data Persistence NFRs.

3. **Event Bus - Responsibilities:** Decoupling critical path from auditing and billing integration. - **Satisfies:** FR-LOG, NFR-PERF.

**Level 2: Component Whitebox¶** (Deferred to Level 3 design)

### **Runtime View¶¶**

This section describes concrete behavior and interactions between building blocks in the form of scenarios.

### **Key Scenarios¶**

**Scenario 1: [Name]¶¶** Todo

Describe the flow, referencing actors (ACT-\*) and components.

**Scenario 2: Error Handling¶¶** Todo

Describe how errors flow through the system, referencing Error Handling.

### **Deployment View¶¶**

This section describes the technical infrastructure where the system executes.

**Infrastructure Level 1¶¶** Todo

Add Deployment Diagram.

### **Mapping to Infrastructure¶¶**

- **[Node Name]:** Hosts [Container/Component].
- **Network:** (Description of network zones/security groups).

**NFR Satisfaction¶¶** This deployment structure supports:

- Availability
- Capacity

### **Cross-cutting Concepts¶¶**

This section describes concepts that apply across multiple building blocks.

**Security / Authentication**¶ See Security NFRs.

Todo

Explain *how* authentication is implemented across services (e.g., JWT propagation).

**Logging and Monitoring**¶ See Monitoring NFRs.

Todo

Explain the centralized logging architecture.

**Configuration Management**¶ Todo

Describe how configuration is managed (env vars, config service, etc.).

**Architecture Constitution**¶

This document defines the rules and conventions specifically for the architecture documentation.

**Relationship to Main Constitution** This constitution inherits all principles from the Main Specification Constitution. In case of conflict, the Main Constitution takes precedence for spec-related matters, while this document governs architectural decisions.

**Architectural Decision Records (ADRs)** - All significant architectural decisions **MUST** be recorded as ADRs in doc/adr. - ADRs **SHOULD** follow the [Madr](https://adr.github.io/madr/) template.

**Diagramming Standards - Format:** Diagrams **SHOULD** be defined using Mermaid.js code blocks where possible for version control. - **Model:** The C4 model (Context, Containers, Components, Code) **SHALL** be used for structural views.

**References** - Architecture views **MUST** reference Functional Requirements (FR) and Non-Functional Requirements (NFR) using their identifiers. - Architecture views **SHOULD NOT** restate requirement text.

**Architectural Views and Their Roles**¶ This project adopts a layered architecture documentation model. Each view has a clearly defined purpose and scope. Views **SHALL NOT** overlap in responsibility.

**Specification (Requirements Layer)**¶

- Functional Requirements (FR), Non-Functional Requirements (NFR), Constraints (CON), and Error Definitions (ERR) define **what the system must do** and **under what conditions**.
- Specification documents are the **single source of truth** for requirements.



- Architecture documents SHALL NOT redefine or restate requirements.

**Context View (C4 Level 1)¶** **Purpose:** - Define the system boundary and its external environment.

**Scope:** - External actors and external systems - High-level interactions across the system boundary

**Rules:** - The Context View SHALL NOT describe internal structure. - Use cases and actors SHALL be referenced, not restated.

**Building Block View (C4 Level 2 / 3)¶** **Purpose:** - Allocate responsibilities to architectural building blocks.

**Scope:** - Major deployable units (containers) and their responsibilities - Relationships between building blocks - Mapping of responsibilities to Functional Requirements (by reference)

**Rules:** - Building Blocks represent **responsibility-bearing units**, not technologies. - Implementation technologies SHALL NOT be specified. - Each Functional Requirement SHALL have a clear architectural owner.

**Runtime View¶** **Purpose:** - Explain how building blocks collaborate at runtime to fulfill key scenarios.

**Scope:** - Representative interaction flows - Responsibility transitions between building blocks - Normal and exceptional execution paths (by reference to ERR)

**Rules:** - Runtime Views SHALL NOT redefine Functional Requirements. - Error semantics are defined in ERR and SHALL be referenced only.

**Architecture Decision Records (ADR)¶** **Purpose:** - Record significant architectural decisions and their rationale.

**Scope:** - Decisions involving trade-offs, alternatives, or long-term impact - Technology choices, architectural patterns, and deployment strategies

**Rules:** - ADRs are created when multiple viable options exist. - Architecture Views SHALL reference ADRs where decisions affect the structure. - ADRs SHALL NOT redefine requirements.

**Deployment View¶** **Purpose:** - Describe how building blocks are deployed onto physical or cloud infrastructure.

**Scope:** - Environments, regions, and network boundaries - Infrastructure components (e.g., cloud services) - Mapping of building blocks to deployment units

**Rules:** - Deployment Views MAY reference specific platforms or services (e.g., AWS). - Deployment Views SHALL demonstrate how NFRs and CONs are satisfied. - No new Functional Requirements SHALL be introduced.

**Cross-cutting Concepts¶** **Purpose:** - Describe architectural mechanisms that apply across multiple building blocks.

**Scope:** - Security, authentication, authorization - Logging, monitoring, and observability - Configuration, multi-tenancy, and data isolation

**Rules:** - Cross-cutting Concepts explain **how** requirements are realized structurally. - Detailed requirement definitions remain in the Specification layer. - Concepts SHALL be consistent across all relevant views.

**Dependency Direction¶** The dependency direction between documents SHALL be strictly maintained:

- Specification → Architecture Views
- Architecture Views → ADRs
- Architecture Views → Deployment Views

Reverse dependencies SHALL NOT exist.

## Architecture Decision Records¶

### Architecture Decision Records¶

This directory contains the Architecture Decision Records (ADRs) for the project.

**Creating a new ADR:** Copy ADR-YYYY-NNN: [Short Title] to a new file named XXXX-title.rst (e.g., 0001-language-selection.rst).

### Indices¶

**ADR Constitution¶** This document defines the rules and conventions for Architecture Decision Records (ADRs).

**Mandatory ADRs¶** Before implementation begins, the following core architectural decisions **MUST** be made and recorded (Minimum 3-5 records ideal):

1. **Language / Runtime Selection:** (e.g., Python, Node.js, Go)
2. **Communication Pattern:** Synchronous (REST/gRPC) vs Asynchronous (Messaging).
3. **State Management:** Stateful vs Stateless architecture.
4. **Data Persistence:** Datastore selection (Relational, NoSQL, etc.).
5. **Authentication:** Concrete authentication method (OIDC, JWT, Session).

**Directory Structure & Naming¶** ADRs are stored in `doc/adr/` and follow a strict naming convention to ensure ordering.

**Structure:**

```
doc/adr/
|-- constitution.rst      # This file (Rules)
|-- index.rst            # Table of Contents
|-- template.rst         # Copy this to create new ADRs
|-- 0001-language.rst    # Sequential ID and kebab-case title
|-- 0002-database.rst
`-- ...
```

**Naming Convention:** - Filename: `NNNN-short-title.rst` - `NNNN`: Monotonic integer, distinct (e.g., 0001, 0002). - `short-title`: Kebab-case description (e.g., `event-bus-selection`).

**Lifecycle¶**

- **Proposed:** Draft state, under review.
- **Accepted:** Approved and binding.
- **Deprecated:** No longer valid due to newer decisions.
- **Superseded:** Replaced by a specific newer ADR (reference the new ID).

**ADR-YYYY-NNN: [Short Title]¶**

- Status: [Proposed | Accepted | Deprecated | Superseded]
- Deciders: [List of people involved]
- Date: [YYYY-MM-DD]

**Context and Problem Statement¶** [Describe the context and problem statement, e.g., in free form using two or three sentences. You may want to articulate the problem in form of a question.]

**Decision Drivers¶**

- [driver 1, e.g., a force, facing concern, ...]
- [driver 2, e.g., a force, facing concern, ...]

**Considered Options¶**

- [Option 1]
- [Option 2]
- [Option 3]

**Decision Outcome¶** Chosen option: “[Option 1]”, because [justification. e.g., only option, which meets k.o. criterion decision driver | which resolves force force | ... | comes out best (see below)].

Positive Consequences¶

- [e.g., improvement of quality attribute satisfaction, follow-up decisions required, ...]

Negative Consequences¶

- [e.g., compromising quality attribute satisfaction, follow-up decisions required, ...]

**Pros and Cons of the Options¶** Option 1¶

[example | description]

- Good, because [argument a]
- Good, because [argument b]
- Bad, because [argument c]

Option 2¶

[example | description]

- Good, because [argument a]
- Bad, because [argument b]

**ADR-0001: Language and Runtime Selection¶**

- Status: Proposed
- Deciders: Project Team
- Date: 2026-01-11

**Context and Problem Statement¶** We need to select the primary programming language and runtime environment for the backend services Control Plane. The choice affects development speed, ecosystem availability, performance, and long-term maintainability.

Decision Drivers¶

- Ecosystem for SaaS / Web APIs
- Performance constraints
- Developer familiarity
- Type safety and maintainability

### Considered Options¶

- Python (FastAPI / Django)
- TypeScript (Node.js / NestJS)
- Go (Gin / Echo)
- Java / Kotlin (Spring Boot)

### Decision Outcome¶ [Proposed]: [Option]

#### Positive Consequences¶

- [Allows rapid development...]

#### Negative Consequences¶

- [Runtime performance lower than...]

### ADR-0002: Service Communication Pattern (Sync vs Async)¶

- Status: Proposed
- Deciders: Project Team
- Date: 2026-01-11

**Context and Problem Statement¶** The system consists of a Control Plane and potentially distributed managed applications or agents. We need to decide how these components communicate, specifically balancing coupling, latency, and reliability.

#### Decision Drivers¶

- Loose coupling between services
- Latency requirements for user-facing actions
- System resilience and handling of downtime
- Complexity of operations

### Considered Options¶

- Synchronous REST/gRPC only
- Asynchronous Messaging (Event-driven) primarily
- Hybrid (Sync for queries, Async for mutations)

**Decision Outcome¶** [Proposed]: [Option]

Positive Consequences¶

- ...

Negative Consequences¶

- ...

### **ADR-0003: State Management (Stateful vs Stateless)¶**

- Status: Proposed
- Deciders: Project Team
- Date: 2026-01-11

**Context and Problem Statement¶** We need to determine if the backend services should maintain session or application state in memory (Stateful) or delegate all state to external stores (Stateless). This impacts scalability and deployment strategies.

Decision Drivers¶

- Horizontal scalability
- Deployment ease (Blue/Green, Canary)
- Complexity of state synchronization
- Resources efficiency

**Considered Options¶**

- Stateless (Recommended for Cloud Native)
- Stateful (Sticky sessions)

**Decision Outcome¶** [Proposed]: Stateless

Positive Consequences¶

- Easy to scale out
- Resilience to instance failures

Negative Consequences¶

- External store dependency (Redis/DB) for temporary state

#### **ADR-0004: Data Persistence Strategy¶**

- Status: Proposed
- Deciders: Project Team
- Date: 2026-01-11

**Context and Problem Statement¶** We need to select the primary data storage technology for the Core Database (Tenants, Users, Config). The data is highly relational and requires strong consistency (ACID) for billing and provisioning critical paths.

##### **Decision Drivers¶**

- ACID compliance
- Relational data integrity (Foreign Keys)
- Cloud provider support (Managed Services)
- Scalability requirements

##### **Considered Options¶**

- PostgreSQL
- MySQL / MariaDB
- NoSQL (MongoDB, DynamoDB)

**Decision Outcome¶** [Proposed]: PostgreSQL

##### **Positive Consequences¶**

- Strong strict consistency
- Rich ecosystem (JSONB support, extensions)

##### **Negative Consequences¶**

- Vertical scaling limits compared to NoSQL

#### **ADR-0005: Authentication Method Implementation¶**

- Status: Proposed
- Deciders: Project Team
- Date: 2026-01-11

**Context and Problem Statement¶** We need to define *how* authentication is implemented. Specifically, the mechanism for transmitting identity and validating it across the Control Plane and Managed Apps.

Decision Drivers¶

- Security standards compliance (OAuth2 / OIDC)
- Statelessness
- Compatibility with SPA / Mobile clients
- Ease of revocation

**Considered Options¶**

- JWT (JSON Web Tokens) - Access/Refresh Token pattern
- Server-side Code Sessions (Cookie-based)
- API Keys (for M2M only)

**Decision Outcome¶** [Proposed]: JWT (Access/Refresh Tokens)

Positive Consequences¶

- Stateless verification by services
- Standardized (OIDC compatible)

Negative Consequences¶

- Token revocation complexity
- Token size