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`.

Category	ID Prefix	Directory
Terminology	TERM	doc/spec/terminology
Functional Requirement	FR	doc/spec/functional-requirements
Non-Functional Req	NFR	doc/spec/non-functional-requirements
Use Case	UC	doc/spec/use-cases
Constraint	CON	doc/spec/constraints-and-assumptions
Data Entity	DATA	doc/spec/data-model
Interface / API	API / IF	doc/spec/interface-requirements
Error Definition	ERR	doc/spec/error-handling
Actor	ACT	doc/spec/actors
Architecture Decision	ADR	doc/adr

- **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)`.

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 <ACT>`. - **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.

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 <ACT-OPS>` accounts (registration, deletion, identity management) and their authentication to the `Control Plane <TERM-SYS-CP>` (`IF-OPS-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

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 <TERM-APP-TARGET>`. Interacts with the system to register applications and manage API credentials.

Authentication & Authorization

FR-AUTH-001 Supported Authentication Methods

The `TERM-SYS-CP` SHALL support the following authentication methods for `Tenant Users <DAT-USER>`:

- OpenID Connect (OIDC)

- Password-based authentication

Realized by: Universal Login Page <IF-LOGIN-UI>

FR-AUTH-003 Tenant SSO Configuration

The TERM-SYS-CP SHALL allow a Tenant Owner (role of ACT-USER, see DAT-ROLE) to register an external Identity Provider (IdP) for Single Sign-On (SSO). The configuration SHALL be stored in SSO Configuration <DAT-SSO-CONFIG>. **Realized by**: Tenant Administration Console <IF-TENANT-CONSOLE>

FR-AUTH-004 Password Reset

The TERM-SYS-CP SHALL allow Tenant Users <DAT-USER> (using password authentication) to request a password reset via their registered email address. The TERM-SYS-CP SHALL allow authenticated Tenant Users <DAT-USER> to change their password.

Realized by: Universal Login Page <IF-LOGIN-UI>

Tenant Administration

FR-TENANT-001 User Invitation

The TERM-SYS-CP SHALL allow Tenant Owners and Administrators to invite new Users <DAT-USER> to their Tenant <DAT-TENANT>. This process SHALL create a User Invitation <DAT-INVITE> record.

Realized by: Tenant Administration Console <IF-TENANT-CONSOLE>

FR-TENANT-002 User Deletion

The TERM-SYS-CP SHALL allow Tenant Owners and Administrators to delete Users <DAT-USER> from their Tenant <DAT-TENANT>.

Realized by: Tenant Administration Console <IF-TENANT-CONSOLE>

FR-TENANT-003 Contract Modification

The TERM-SYS-CP SHALL allow only Tenant Owners to modify the tenant's subscription contract (specifically Tenant.plan <DAT-TENANT>).

Realized by: Tenant Administration Console <IF-TENANT-CONSOLE>

FR-TENANT-004 User Role Management

The TERM-SYS-CP SHALL allow Tenant Owners and Administrators to modify the Roles <DAT-ROLE> of existing Users <DAT-USER> within their Tenant <DAT-TENANT>.

Realized by: Tenant Administration Console <IF-TENANT-CONSOLE>

Platform Operations

FR-OPS-001 Tenant Status Management

The TERM-SYS-CP SHALL allow ACT-OPS to modify the status of a Tenant <DAT-TENANT> (e.g., Active, Suspended). When a Tenant <DAT-TENANT> is Suspended, the system SHALL revoke access for all Users <DAT-USER> associated with that tenant.

Realized by: Operator Console <IF-OPS-CONSOLE>

System Operations

FR-SYS-001 Application Registration

The TERM-SYS-CP SHALL allow ACT-DEV to register a new Managed Application <DAT-APP>. The system SHALL generate a unique Application ID upon registration.

FR-SYS-002 API Key Management

The TERM-SYS-CP SHALL allow ACT-DEV to issue API Access Keys <DAT-KEY> for a registered Managed Application <DAT-APP>. The system SHALL display the Client Secret only once upon issuance. The system SHALL allow ACT-DEV to revoke existing keys.

Feature Flag Management

FR-FLAG-001 Flag Configuration

The TERM-SYS-CP SHALL allow ACT-OPS to configure Feature Flags <DAT-FLAG> for each Tenant <DAT-TENANT>.

Realized by: Operator Console <IF-OPS-CONSOLE>

FR-FLAG-002 Flag Delivery

The system SHALL provide an interface via API-FLAG for TERM-APP-TARGET to retrieve the current state of Feature Flags <DAT-FLAG>.

Realized by: API-FLAG

Billing & Usage

FR-BILL-001 Billing Event Persistence

The TERM-SYS-CP SHALL persistently record billable events triggers received via FR-BILL-002 as Billing Events <DAT-BILL-EVENT>.

Realized by: API-BILL

FR-BILL-002 Billing Event Ingestion

The system SHALL provide an API (API-BILL) that allows TERM-APP-TARGET to report billable events (corresponding to Billing Events <DAT-BILL-EVENT>).

Realized by: API-BILL

Audit & Logging

FR-LOG-001 Audit Log Collection

The TERM-SYS-CP SHALL collect security and operational logs from all components via API-LOG and persist them as Audit Logs <DAT-LOG>.

Realized by: API-LOG

FR-LOG-002 Audit Log Export

The TERM-SYS-CP SHALL allow ACT-AUDIT to export Audit Logs <DAT-LOG> in CSV format.

Realized by: Auditor Console <IF-AUDIT-CONSOLE>

FR-LOG-003 Control Plane Auditing

The TERM-SYS-CP SHALL record its own state-changing operations (e.g., Tenant Provisioning, User Management) as Audit Logs <DAT-LOG>.

Tenant Provisioning & Lifecycle

UC-PROV-001 Tenant Provisioning

Actor: Platform Operator <ACT-OPS>

Description: The Platform Operator <ACT-OPS> creates a new tenant configuration and enables subscribed features, allowing the tenant to immediately access the Managed Application <TERM-APP-TARGET>.

Trigger: A new customer subscription is confirmed.

Preconditions:

1. The Platform Operator <ACT-OPS> is logged in.

Postconditions:

1. A new tenant entity is created in the Control Plane <TERM-SYS-CP>.
2. Initial Tenant User <ACT-USER> (Role: Owner) is provisioned.
3. Feature flags corresponding to the subscription plan are active.

Scenario:

1. The Platform Operator <ACT-OPS> navigates to the **Operator Console**.
2. The Platform Operator <ACT-OPS> enters tenant details (Name, Domain, Plan) and the email address for the initial Owner.
3. The Platform Operator <ACT-OPS> selects the Managed Application <TERM-APP-TARGET> to enable.
4. The Platform Operator <ACT-OPS> selects the "Provision" action.
5. The Control Plane <TERM-SYS-CP> creates the tenant and the initial Tenant User <ACT-USER> (Role: Owner).
6. The Control Plane <TERM-SYS-CP> enables access to the Managed Application <TERM-APP-TARGET>.

Related Requirements:

- Flag Configuration <FR-FLAG-001>
- User Invitation <FR-TENANT-001>
- Contract Modification <FR-TENANT-003>
- Control Plane Auditing <FR-LOG-003>

UC-TENANT-SUSPEND Tenant Suspension

Actor: Platform Operator <ACT-OPS>

Description: The Platform Operator <ACT-OPS> suspends a tenant's access to the managed application, usually due to non-payment or policy violation.

Trigger: The Platform Operator <ACT-OPS> selects "Suspend Tenant" in the **Operator Console**.

Preconditions:

1. The Platform Operator <ACT-OPS> is logged in.
2. The target tenant is currently Active.

Postconditions:

1. The Tenant <DAT-TENANT> status is updated to Suspended.

2. All Users <DAT-USER> under the tenant are immediately denied access.

Scenario:

1. The Platform Operator <ACT-OPS> searches for the tenant in the **Operator Console**.
2. The Platform Operator <ACT-OPS> selects the "Suspend" action.
3. The Platform Operator <ACT-OPS> provides a reason (optional).
4. The Platform Operator <ACT-OPS> confirms the action.
5. The Control Plane <TERM-SYS-CP> invokes the suspension logic.

Related Requirements:

- Tenant Status Management <FR-OPS-001>
- Control Plane Auditing <FR-LOG-003>

Tenant Administration

UC-TENANT-USER-DELETE User Deletion

Actor: Tenant User <ACT-USER> (Role: Owner, Administrator)

Description: The Tenant User <ACT-USER> (Role: Owner or Administrator) removes a user from the tenant organization.

Trigger: The Tenant User <ACT-USER> selects "Delete User" in the **Tenant Administration Console**.

Preconditions:

1. The Tenant User <ACT-USER> is logged in with sufficient privileges.
2. Target user exists.

Postconditions:

1. Target user is removed from authentication and cannot access applications.

Scenario:

1. The Tenant User <ACT-USER> selects the user to remove.
2. The Tenant User <ACT-USER> confirms deletion.
3. The Control Plane <TERM-SYS-CP> removes the user.

Related Requirements:

- User Deletion <FR-TENANT-002>
- Control Plane Auditing <FR-LOG-003>

UC-TENANT-USER-UPDATE User Role Update

Actor: Tenant User <ACT-USER> (Role: Owner, Administrator)

Description: The Tenant User <ACT-USER> (Role: Owner or Administrator) modifies the role of an existing user within the tenant organization.

Trigger: The Tenant User <ACT-USER> selects "Edit Role" in the **Tenant Administration Console**.

Preconditions:

1. The Tenant User <ACT-USER> is logged in with sufficient privileges.
2. Target user exists.

Postconditions:

1. Target user's role is updated.
2. Target user's permissions are immediately adjusted.

Scenario:

1. The `Tenant User <ACT-USER>` selects the user to update.
2. The `Tenant User <ACT-USER>` selects the new role.
3. The `Tenant User <ACT-USER>` saves the changes.
4. The `Control Plane <TERM-SYS-CP>` validates the permissions (e.g., cannot downgrade own role if last Owner).
5. The `Control Plane <TERM-SYS-CP>` updates the user record.

Related Requirements:

- `User Role Management <FR-TENANT-004>`
- `Control Plane Auditing <FR-LOG-003>`

UC-TENANT-INVITE User Invitation

Actor: `Tenant User <ACT-USER>` (Role: Owner, Administrator)

Description: The `Tenant User <ACT-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 <ACT-USER>` selects "Invite User" in the **Tenant Administration Console**.

Preconditions:

1. The `Tenant User <ACT-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 <ACT-USER>` enters the email address and role (Admin or User) of the new user.
2. The `Tenant User <ACT-USER>` submits the invitation.
3. The `Control Plane <TERM-SYS-CP>` validates the input and permissions.
4. The `Control Plane <TERM-SYS-CP>` sends the invitation email.

Related Requirements:

- `User Invitation <FR-TENANT-001>`
- `Control Plane Auditing <FR-LOG-003>`

UC-TENANT-SSO SSO Configuration

Actor: `Tenant User <ACT-USER>` (Role: Owner)

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

Trigger: The `Tenant User <ACT-USER>` initiates "SSO Setup" in the **Tenant Administration Console**.

Preconditions:

1. The `Tenant User <ACT-USER>` is logged in with Owner role.
2. The `Tenant User <ACT-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 <ACT-USER>` enters IdP details (Issuer URL, Client ID, Client Secret).
2. The `Control Plane <TERM-SYS-CP>` verifies the IdP configuration (discovery).
3. The `Control Plane <TERM-SYS-CP>` saves the configuration.

Related Requirements:

- `Tenant SSO Configuration <FR-AUTH-003>`
- `Control Plane Auditing <FR-LOG-003>`

Access Management

UC-LOGIN Tenant User Login

Actor: `Tenant User <ACT-USER>`

Description: The `Tenant User <ACT-USER>` logs in to the system or a managed application using their credentials or an external IdP.

Trigger: The `Tenant User <ACT-USER>` attempts to access a protected resource.

Preconditions:

1. The `Tenant User <ACT-USER>` account exists and is active.

Postconditions:

1. The `Tenant User <ACT-USER>` receives an authentication token.
2. The `Tenant User <ACT-USER>` gains access to the application.

Scenario:

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

Related Requirements:

- `Supported Authentication Methods <FR-AUTH-001>`

UC-AUTH-RESET Password Reset

Actor: `Tenant User <ACT-USER>`

Description: The `Tenant User <ACT-USER>` initiates a password reset flow when they have forgotten their credentials.

Trigger: The `Tenant User <ACT-USER>` selects "Forgot Password" on the **Universal Login Page**.

Preconditions:

1. The `Tenant User <ACT-USER>` has a registered account with an email address.
2. The account is configured for password authentication.

Postconditions:

1. The `Tenant User <ACT-USER>` has updated their credential.
2. The `Tenant User <ACT-USER>` can log in with the new password.

Scenario:

1. The Tenant User <ACT-USER> enters their email address on the **Universal Login Page**.
2. The Control Plane <TERM-SYS-CP> sends a password reset link/token to the email.
3. The Tenant User <ACT-USER> clicks the link and enters a new password.
4. The Control Plane <TERM-SYS-CP> updates the credential store.

Related Requirements:

- Password Reset <FR-AUTH-004>
- Control Plane Auditing <FR-LOG-003>

System Deployment

UC-DEV-REGISTER Application Registration

Actor: Developer <ACT-DEV>

Description: The Developer <ACT-DEV> registers the Managed Application <TERM-APP-TARGET> with the Control Plane to obtain credentials for API access. This is typically done as part of the initial system deployment.

Trigger: The Developer <ACT-DEV> initiates the "Register System" workflow (via CLI or script).

Preconditions:

1. The Developer <ACT-DEV> has administrative access to the Control Plane infrastructure.

Postconditions:

1. A Managed Application <DAT-APP> record is created.
2. An API Access Key <DAT-KEY> is issued and returned to the Developer <ACT-DEV>.
3. The Managed Application <TERM-APP-TARGET> is configured with the key.

Scenario:

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

Related Requirements:

- Application Registration <FR-SYS-001>
- API Key Management <FR-SYS-002>

Audit Management

UC-AUDIT-EXPORT Audit Log Export

Actor: Auditor <ACT-AUDIT>

Description: The Auditor <ACT-AUDIT> exports system audit logs for compliance review.

Trigger: The Auditor <ACT-AUDIT> selects "Export Logs" within the **Auditor Console**.

Preconditions:

1. The Auditor <ACT-AUDIT> is logged in with Auditor privileges.

Postconditions:

1. A CSV file containing the requested logs is downloaded to the Auditor <ACT-AUDIT>'s device.

Scenario:

1. The `Auditor <ACT-AUDIT>` navigates to the "Audit Logs" view in the **Auditor Console**.
2. The `Auditor <ACT-AUDIT>` selects the date range and filters for the export.
3. The `Auditor <ACT-AUDIT>` initiates the download.
4. The `Control Plane <TERM-SYS-CP>` queries the log storage.
5. The `Control Plane <TERM-SYS-CP>` formats the data as CSV and streams the response.

Related Requirements:

- `Audit Log Collection <FR-LOG-001>`
- `Audit Log Export <FR-LOG-002>`

UC-AUDIT-RECORD-CP Control Plane Event Recording

Actor: `Control Plane <TERM-SYS-CP>`

Description: The `Control Plane <TERM-SYS-CP>` 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 <TERM-SYS-CP>` executes the detailed business logic.
3. The `Control Plane <TERM-SYS-CP>` generates an audit log event containing Actor ID, Action, Resource, and Timestamp.
4. The `Control Plane <TERM-SYS-CP>` persists the log entry.

Related Requirements:

- `Control Plane Auditing <FR-LOG-003>`

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 <TERM-APP-TARGET>` (System)

Description: The `Managed Application <TERM-APP-TARGET>` retrieves the active feature flags for its tenant to enable/disable functionality dynamically.

Trigger: The `Managed Application <TERM-APP-TARGET>` starts up or periodic refresh interval elapses.

Postconditions:

1. The `Managed Application <TERM-APP-TARGET>` receives the current set of flags.

Scenario:

1. The `Managed Application <TERM-APP-TARGET>` requests flags from the `Control Plane <TERM-SYS-CP>`.
2. The `Control Plane <TERM-SYS-CP>` identifies the tenant (via keys or context).

3. The Control Plane <TERM-SYS-CP> returns the flag configuration.

Related Requirements:

- Flag Delivery <FR-FLAG-002>

UC-SYS-BILL-REPORT Billing Event Reporting

Actor: Managed Application <TERM-APP-TARGET> (System)

Description: The Managed Application <TERM-APP-TARGET> reports a billable event to the Control Plane <TERM-SYS-CP> for tracking.

Trigger: A user performs a billable action within the Managed Application <TERM-APP-TARGET>.

Postconditions:

1. The event is persisted in the Control Plane <TERM-SYS-CP>.

Scenario:

1. The Managed Application <TERM-APP-TARGET> detects billable event.
2. The Managed Application <TERM-APP-TARGET> sends event data to the Control Plane <TERM-SYS-CP>.
3. The Control Plane <TERM-SYS-CP> validates and stores the event.

Related Requirements:

- Billing Event Persistence <FR-BILL-001>
- Billing Event Ingestion <FR-BILL-002>

UC-SYS-LOG-REPORT Audit Log Reporting

Actor: Managed Application <TERM-APP-TARGET> (System)

Description: The Managed Application <TERM-APP-TARGET> sends its security and operation logs to the Control Plane <TERM-SYS-CP> for centralized auditing.

Trigger: The Managed Application <TERM-APP-TARGET> generates a log entry.

Postconditions:

1. The log entry is collected by the Control Plane <TERM-SYS-CP>.

Scenario:

1. The Managed Application <TERM-APP-TARGET> generates a log.
2. The Managed Application <TERM-APP-TARGET> streams/sends the log to the Control Plane <TERM-SYS-CP>.
3. The Control Plane <TERM-SYS-CP> ingests the log.

Related Requirements:

- Audit Log Collection <FR-LOG-001>

Schema Definitions

This section defines the core data entities managed by the Control Plane <TERM-SYS-CP>.

Tenants

Represents a customer organization subscribed to the `Managed Application <TERM-APP-TARGET>`.

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., `acme`).
plan	Enum	Subscription plan (e.g., `Free`, `Pro`, `Enterprise`).
status	Enum	Account status (`Active`, `Suspended`).
created_at	Timestamp	Record creation time.

Users

Represents an individual user belonging to a `Tenant <DAT-TENANT>` or the Platform.

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

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 <TERM-APP-TARGET>` features only.
- **Operator**: (Platform level) Full access to the `Control Plane <TERM-SYS-CP>`.

Feature Flags

Controls the availability of features for specific tenants.

Field	Type	Description
id	UUID	Unique identifier.
tenant_id	UUID	Foreign Key to `Tenants <DAT-TENANT>`.
key	String	Feature identifier (e.g., `ai_module_enabled`).
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.

Field	Type	Description
actor_type	Enum	Type of actor (`User`, `Operator`, `System`).
action	String	Description of the operation (e.g., `tenant.create`).
resource	String	Identifier of the target resource.
outcome	Enum	Result of the operation (`Success`, `Failure`).
metadata	JSON	Additional context (e.g., previous values, IP address).

Managed Applications

Represents a registered `Managed Application <TERM-APP-TARGET>` 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 <ACT-DEV>` or owner.
status	Enum	Registration status (`Active`, `Revoked`).

API Access Keys

Credentials used by `Managed Applications <DAT-APP>` to authenticate against `Control Plane <TERM-SYS-CP>` APIs.

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

SSO Configuration

Stores the Identity Provider details for a `Tenant <DAT-TENANT>`.

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

Billing Events

Raw records of billable activities reported by applications.

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

User Interfaces

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

IF-OPS-CONSOLE Operator Console

User: `ACT-OPS` **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` (Owner, Admin) **Description**: The self-service web portal for Tenant Administrators. Allows management of users, invitations, SSO configuration, and subscription viewing.

IF-AUDIT-CONSOLE Auditor Console

User: `ACT-AUDIT` **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`, `ACT-OPS`, `ACT-AUDIT` **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**: Input (`TERM-APP-TARGET` -> `TERM-SYS-CP`) **Purpose**: To report billable operations performed within the managed SaaS applications. **Payload**: SHALL include Tenant ID, Timestamp, Event Type, and Quantity.

API-LOG Audit Log API

Type: REST API **Direction**: Input (`TERM-APP-TARGET` -> `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 (`TERM-SYS-CP` -> `TERM-APP-TARGET`) **Purpose**: To retrieve the active feature flags for a specific tenant. **Caching**: Managed apps SHOULD cache this response to minimize latency.