EC3L Platform — Architecture & Workflow Reference

Version: Phase 26 (2026-02-21) **Purpose:** Comprehensive platform reference for AI-assisted UI/UX design **Stack:** React 19 + Vite + TanStack Query | Express + Drizzle ORM + PostgreSQL | Vitest

---

Table of Contents

---

1. Platform Overview

EC3L is a **stateless, multi-tenant control plane** for managing enterprise configuration changes with deterministic execution, audit trails, and governance gates. It is NOT a ServiceNow clone — it is an execution-oriented platform where changes are validated and applied in place, not promoted as XML between instances.

Core Primitives

Primitive	Purpose
Tenant	Isolated organization boundary (slug-based, server-resolved)
Project	Scoping container for record types, changes, environments
Record Type	Schema definition (typed fields, inheritance via baseType)
Change Record	Unit of work with lifecycle (Draft → Merged)
Change Target	Mutation scope declaration (record_type, form, workflow, file, etc.)
Patch Op	Atomic schema mutation (set_field, add_field, remove_field, rename_field)
Snapshot	Pre-mutation record of schema state (audit trail)
Module	Code/schema/workflow organizational unit
Environment	Deployment target (dev, test, prod) with governance gates
Workflow	Multi-step automation triggered by events or manual action

Record Instance	Runtime data record created from a record type
Graph Package	Installable bundle of record types, SLA policies, assignment rules, workflows
Vibe Draft	AI-generated package draft with versioning and inline editing
Promotion Intent	Governed promotion request with approval lifecycle

Architecture Layers

---

2. Multi-Tenant Architecture

Resolution Flow

1. **Client** stores tenant slug in localStorage, sends as `x-tenant-id` header on every request
2. **Middleware** resolves slug → UUID (server-side only), rejects unknown slugs (404), missing header (401)
3. **TenantStorage** closure captures `tenantId`, ensuring every query filters by tenant
4. **Executor** cross-validates project consistency (rt.projectId === change.projectId)

Identity Headers

Header	Purpose	Required
`x-tenant-id`	Tenant slug (resolved server-side to UUID)	Yes (all /api/* except GET /api/tenants)
`x-user-id`	Human user identifier	Optional
`x-agent-id`	Agent identifier (mutually exclusive with x-user-id)	Optional

Security Guarantees

- **No sessions, cookies, or client-supplied tenant UUIDs** (T6)
- Tenant ID resolved server-side from slug (T1)
- All data access tenant-scoped via WHERE tenant_id = ctx.tenantId (T2)
- Unknown slug → 404 (not 401, to prevent enumeration) (T4)
- Cross-tenant data masked as 404 (never 403, no info leak)

---

3. Database Schema (Complete)

Enums

Change Management

- **changeStatusEnum:** Draft, Implementing, WorkspaceRunning, Validating, ValidationFailed, Ready, Merged
- **changeEventTypeEnum:** change_status_changed, change_target_added, change_target_deleted, patch_op_added, patch_op_deleted, environment_release_created, environment_deployed
- **changeTargetTypeEnum:** form, workflow, rule, record_type, script, file
- **promotionIntentStatusEnum:** draft, previewed, approved, executed, rejected

Workflow

- **wfTriggerTypeEnum:** record_event, schedule, manual
- **wfDefinitionStatusEnum:** draft, active, retired
- **wfStepTypeEnum:** assignment, approval, notification, decision, record_mutation, record_lock
- **wfExecutionStatusEnum:** running, paused, completed, failed
- **wfStepExecutionStatusEnum:** pending, awaiting_approval, completed, failed
- **wfTriggerStatusEnum:** active, disabled
- **wfIntentStatusEnum:** pending, running, completed, dispatched, failed, duplicate

Data & Configuration

- **recordTypeStatusEnum:** draft, active, retired
- **fieldTypeEnum:** string, number, boolean, date, datetime, reference, choice, text
- **choiceListStatusEnum:** active, disabled
- **formDefinitionStatusEnum:** draft, active, retired
- **formBehaviorRuleTypeEnum:** visible, required, readOnly
- **recordTimerStatusEnum:** pending, breached, completed
- **vibePackageDraftStatusEnum:** draft, previewed, installed, discarded

Environment

- **environmentNameEnum:** dev, test, prod
- **environmentReleaseStatusEnum:** created
- **moduleTypeEnum:** code, schema, workflow, ui, integration, application
- **capabilityProfileEnum:** CODE_MODULE_DEFAULT, WORKFLOW_MODULE_DEFAULT, READ_ONLY

Template & Installation

- **templateDomainEnum:** HR, Finance, Legal, Facilities, Custom, ITSM
- **installedAppStatusEnum:** installed, upgrading, failed
- **installEventTypeEnum:** install_started, install_completed, install_failed

RBAC & Agents

- **rbacActorTypeEnum:** user, agent, system
- **rbacRoleStatusEnum:** active, disabled
- **rbacPolicyEffectEnum:** allow, deny
- **rbacAuditOutcomeEnum:** allow, deny
- **rbacResourceTypeEnum:** form, workflow, override, change
- **agentProposalTypeEnum:** form_patch, workflow_change, approval_comment
- **agentProposalStatusEnum:** draft, submitted, accepted, rejected
- **agentRunStatusEnum:** Pending, Running, Passed, Failed
- **overrideTypeEnum:** workflow, form, rule, config
- **overrideStatusEnum:** draft, active, retired

Telemetry

- **telemetryEventTypeEnum:** 50+ event types (see Domain Event System section)
- **telemetryExecutionTypeEnum:** workflow_step, task, agent_action
- **telemetryActorTypeEnum:** user, agent, system
- **workspaceStatusEnum:** Pending, Running, Stopped, Failed

Tables by Domain

Core: Tenants & Projects

Table	Key Columns	Notes
tenants	id, name, slug (unique), plan, createdAt	Organization boundary
projects	id, name, githubRepo, defaultBranch, description, tenantId→tenants	Scoping container

Modules & Changes

Table	Key Columns	Notes
modules	id, projectId→projects, name, type, rootPath, version, capabilityProfile	Organizational unit
changeRecords	id, projectId→projects, title, description, baseSha, modulePath, moduleId→modules, status, branchName, environmentId→environments	Unit of work
workspaces	id, changeId→changeRecords, containerId, previewUrl, status	Dev workspace
agentRuns	id, changeId→changeRecords, intent, skillsUsed (JSON), logs (JSON), status	Agent execution

Change Management

Table	Key Columns	Notes
changeTargets	id, tenantId, projectId, changeId→changeRecords, type, selector (JSON)	Mutation scope
changePatchOps	id, tenantId, changeId→changeRecords, targetId→changeTargets, opType, payload (JSON), previousSnapshot (JSON), executedAt	Atomic mutation
changeEvents	id, tenantId, projectId, changeId→changeRecords, eventType, payload (JSON)	Change audit
recordTypeSnapshots	id, tenantId, projectId (NOT NULL), recordTypeKey, changeId→changeRecords, schema (JSON)	Pre-mutation audit. Unique: (changeId, recordTypeKey)

Environments & Releases

Table	Key Columns	Notes
environments	id, projectId→projects, name (dev/test/prod), isDefault, requiresPromotionApproval, promotionWebhookUrl	Deployment target
environmentReleases	id, projectId, environmentId→environments, createdBy, status	Release snapshot
environmentReleaseChanges	releaseId→environmentReleases, changeId→changeRecords	Release-change mapping
environmentDeployments	id, projectId, environmentId, releaseId, promotedFromReleaseId	Deployment history

Record Types & Data Dictionary

Table	Key Columns	Notes
recordTypes	id, tenantId, projectId (NOT NULL), name (unique/tenant), key (unique/tenant+project), baseType, schema (JSON), assignmentConfig (JSON), slaConfig (JSON), version, status	Schema definition

fieldDefinitions	id, recordTypeId, name, label, fieldType, isRequired, defaultValue (JSON), choiceListId, referenceRecordTypeId, orderIndex	Field metadata
choiceLists	id, tenantId, name (unique/tenant), status	Dropdown option sets
choiceItems	id, choiceListId, value, label, orderIndex, isActive	Individual options

Record Instances & Timers

Table	Key Columns	Notes
recordInstances	id, tenantId, recordTypeId→recordTypes, data (JSON), createdBy, assignedTo, assignedGroup, createdAt, updatedAt	Runtime data
recordTimers	id, tenantId, recordId→recordInstances, type, dueAt, status (pending/breached/completed), createdAt, updatedAt	SLA tracking
recordLocks	id, tenantId, recordTypeId, recordId, lockedBy, workflowExecutionId, reason. Unique: (tenantId, recordTypeId, recordId)	Concurrency control

Forms & Behavior

Table	Key Columns	Notes
formDefinitions	id, tenantId, recordTypeId, name, version, status. Unique: (tenantId, recordTypeId, name)	Form layout
formSections	id, formDefinitionId, title, orderIndex	Section grouping
formFieldPlacements	id, formSectionId, fieldDefinitionId, column, orderIndex	Field positioning
formBehaviorRules	id, formDefinitionId, ruleType (visible/required/readOnly), targetFieldDefinitionId, condition (JSON), value, orderIndex, status	Dynamic behavior

Workflows & Execution

Table	Key Columns	Notes
workflowDefinitions	id, tenantId, name, triggerType, triggerConfig (JSON), version, status, changeId	Definition
workflowSteps	id, workflowDefinitionId, stepType, config (JSON), orderIndex	Step sequence
workflowExecutions	id, tenantId, workflowDefinitionId, intentId, status, input (JSON), accumulatedInput (JSON), pausedAtStepId, error, startedAt, completedAt	Runtime
workflowStepExecutions	id, workflowExecutionId, workflowStepId, status, output (JSON), executedAt	Step results
workflowTriggers	id, tenantId, workflowDefinitionId, triggerType, triggerConfig (JSON), status	Event bindings
workflowExecutionIntents	id, tenantId, workflowDefinitionId, triggerType, triggerPayload (JSON), idempotencyKey, status, executionId, error, createdAt, dispatchedAt	Async dispatch

Templates & Installation

Table	Key Columns	Notes

templates	id, name, domain, version, description, isGlobal	System catalog
templateModules	id, templateId, moduleType, moduleName, defaultCapabilityProfile, orderIndex, metadata (JSON)	Module definitions
installedApps	id, tenantId, templateId, templateVersion, status, installedAt. Unique: (tenantId, templateId)	Installation state
installedModules	id, installedAppId, moduleId, templateModuleId, capabilityProfile, isOverride	Module mapping
installedAppEvents	id, installedAppId, templateId, tenantId, eventType, errorDetails	Install audit
moduleOverrides	id, tenantId, installedModuleId, overrideType, targetRef, patch (JSON), createdBy, version, status, changeId	Config overrides

RBAC & Audit

Table	Key Columns	Notes
rbacPermissions	id, name (unique), description	Permission catalog
rbacRoles	id, tenantId, name (unique/tenant), description, status	Tenant roles
rbacRolePermissions	roleId, permissionId	Role-permission mapping
rbacUserRoles	userId, roleId	User-role assignment
rbacPolicies	id, tenantId, roleId, resourceType, resourceId, effect (allow/deny)	Fine-grained policies
rbacAuditLogs	id, tenantId, actorType, actorId, permission, resourceType, resourceId, outcome, reason, timestamp	Audit trail

Agent Features

Table	Key Columns	Notes
agentProposals	id, tenantId, changeId, agentId, proposalType, targetRef, payload (JSON), summary, status	Agent-generated proposals

Telemetry

Table	Key Columns	Notes
executionTelemetryEvents	id, eventType, timestamp, tenantId, moduleId (NOT NULL), executionType, workflowId, workflowStepId, executionId, actorType, actorId, status, errorCode, errorMessage, affectedRecordIds (JSON)	Observable events

Graph Packages & Promotion

Table	Key Columns	Notes
graphPackageInstalls	id, tenantId, projectId, packageKey, version, checksum, installedBy, installedAt, diff (JSON), packageContents (JSON)	Install audit trail
environmentPackageInstalls	id, tenantId, projectId, environmentId, packageKey, version, checksum, installedBy, installedAt, source ("install"/"promote"), diff (JSON), packageContents (JSON)	Environment-scoped state

promotionIntents	id, tenantId, projectId, fromEnvironmentId, toEnvironmentId, status, createdBy, createdAt, approvedBy, approvedAt, diff (JSON), result (JSON), notificationStatus, notificationLastError, notificationLastAttemptAt	Governed promotion

Vibe Authoring

Table	Key Columns	Notes
vibePackageDrafts	id, tenantId, projectId, environmentId, status, createdBy, createdAt, updatedAt, prompt, package (JSON), checksum, lastPreviewDiff (JSON), lastPreviewErrors (JSON)	Draft persistence
vibePackageDraftVersions	id, tenantId, draftId, versionNumber, createdAt, createdBy, reason, package (JSON), checksum, previewDiff (JSON), previewErrors (JSON). Unique: (tenantId, draftId, versionNumber)	Version history

---

4. API Surface (Complete)

Global Rules

- All `/api/*` routes require `x-tenant-id` header (except GET /api/tenants)
- Content-Type: application/json for POST/PUT/PATCH
- Error responses: 400 (validation), 401 (missing identity), 403 (RBAC/agent guard), 404 (not found), 409 (state conflict), 422 (execution failure)

Tenants

Method	Path	Purpose
GET	/api/tenants	List all tenants (no auth required)

Projects

Method	Path	Purpose
GET	/api/projects	List tenant projects
GET	/api/projects/:id	Get single project
POST	/api/projects	Create project (auto-creates 3 environments + default module)
GET	/api/projects/:id/changes	Get changes for project
GET	/api/projects/:id/modules	Get modules for project
GET	/api/projects/:id/environments	Get environments for project

Changes

Method	Path	Purpose	Guards
GET	/api/changes	List all changes	Tenant
POST	/api/changes	Create change (Draft)	Tenant
GET	/api/changes/:id	Get change	Tenant
POST	/api/changes/:id/status	Update status	RBAC: change.approve, Agent guard for Ready/Merged
GET	/api/changes/:id/project	Get parent project	Tenant

GET	/api/changes/:id/workspace	Get workspace	Tenant
GET	/api/changes/:id/agent-runs	Get agent runs	Tenant

Change Targets & Patch Ops

Method	Path	Purpose	Guards
POST	/api/changes/:id/targets	Add target (Draft only)	Tenant
GET	/api/changes/:id/targets	List targets	Tenant
POST	/api/changes/:id/patch-ops	Create patch op	Tenant, duplicate guard
GET	/api/changes/:id/patch-ops	List ops	Tenant
DELETE	/api/changes/:id/patch-ops/:opId	Delete pending op	Tenant, execution/status guards

Execution

Method	Path	Purpose	Guards
POST	/api/changes/:id/execute	Dry-run (no status change)	Tenant
POST	/api/changes/:id/merge	Execute + Merged (terminal)	RBAC: change.approve, Agent guard
POST	/api/changes/:id/checkin	Transition to Ready	RBAC: change.approve
POST	/api/changes/:id/start-workspace	Start workspace	Tenant
POST	/api/changes/:id/agent-run	Create agent run	Tenant

Record Types

Method	Path	Purpose
GET	/api/record-types	List all
POST	/api/record-types	Create
GET	/api/record-types/:id	Get by ID
GET	/api/record-types/by-key/:key	Get by key
POST	/api/record-types/:id/activate	Activate
POST	/api/record-types/:id/retire	Retire
GET	/api/record-types/:id/fields	List fields
POST	/api/record-types/:id/fields	Add field

Record Instances

Method	Path	Purpose
POST	/api/record-instances	Create (with auto-assignment + SLA)
GET	/api/record-instances	List (filtered by recordTypeId)
GET	/api/record-instances/:id	Get single
PATCH	/api/record-instances/:id	Update data

Choice Lists

Method	Path	Purpose
GET	/api/choice-lists	List all
POST	/api/choice-lists	Create
GET	/api/choice-lists/:id	Get single
GET	/api/choice-lists/:id/items	List items
POST	/api/choice-lists/:id/items	Add item

Form Definitions

Method	Path	Purpose	Guards
GET	/api/form-definitions	List all	Tenant
POST	/api/form-definitions	Create	Tenant
GET	/api/form-definitions/:id	Get single	Tenant
POST	/api/form-definitions/:id/activate	Activate	Tenant
POST	/api/form-definitions/:id/retire	Retire	Tenant
GET	/api/form-definitions/:id/sections	List sections	Tenant
POST	/api/form-definitions/:id/sections	Add section	Tenant
GET	/api/form-sections/:id/placements	List placements	Tenant
POST	/api/form-sections/:id/placements	Add placement	Tenant
GET	/api/form-definitions/:id/rules	List behavior rules	Tenant
POST	/api/form-definitions/:id/rules	Add rule	Tenant
GET	/api/forms/:recordTypeName/:formName/compiled	Get compiled form	Tenant
POST	/api/forms/:recordTypeName/:formName/overrides	Save form overrides	RBAC: form.edit
POST	/api/forms/:recordTypeName/:formName/vibePatch	Generate AI patch	Tenant

Workflow Definitions

Method	Path	Purpose	Guards
GET	/api/workflow-definitions	List all	Tenant
POST	/api/workflow-definitions	Create (auto-creates linked change)	Tenant
GET	/api/workflow-definitions/:id	Get single	Tenant
POST	/api/workflow-definitions/:id/activate	Activate	Tenant
POST	/api/workflow-definitions/:id/retire	Retire	Tenant
GET	/api/workflow-definitions/:id/steps	List steps	Tenant
POST	/api/workflow-definitions/:id/steps	Add step (Draft only)	Tenant

POST	/api/workflow-definitions/:id/execute	Manual execute	RBAC: workflow.execute, Agent guard
GET	/api/workflow-definitions/:id/triggers	Get triggers	Tenant

Workflow Executions & Triggers

Method	Path	Purpose	Guards
GET	/api/workflow-executions	List all	Tenant
GET	/api/workflow-executions/:id	Get single	Tenant
POST	/api/workflow-executions/:id/resume	Resume paused	RBAC: workflow.approve, Agent guard
GET	/api/workflow-executions/:id/steps	Get step executions	Tenant
GET	/api/workflow-triggers	List all	Tenant
POST	/api/workflow-triggers	Create	Tenant
GET	/api/workflow-triggers/:id	Get single	Tenant
POST	/api/workflow-triggers/:id/enable	Enable	Tenant
POST	/api/workflow-triggers/:id/disable	Disable	Tenant
POST	/api/workflow-triggers/:id/fire	Manual fire	Agent guard

Record Events & Intents

Method	Path	Purpose
POST	/api/record-events	Emit record event
GET	/api/workflow-intents	List intents
GET	/api/workflow-intents/:id	Get single
POST	/api/workflow-intents/dispatch	Dispatch all pending

Environments & Releases

Method	Path	Purpose	Guards
GET	/api/environments/:id	Get environment	Tenant
POST	/api/environments/:id/release	Create release	RBAC: environment.release_create

Templates & Installation

Method	Path	Purpose
GET	/api/templates	List all (system)
GET	/api/templates/:id	Get single
GET	/api/templates/:id/modules	Get modules
POST	/api/templates/:id/install	Install template
GET	/api/installed-apps	List installed apps

Module Overrides

Method	Path	Purpose	Guards
GET	/api/overrides	List all	Tenant
POST	/api/overrides	Create	Tenant
GET	/api/overrides/:id	Get single	Tenant
POST	/api/overrides/:id/activate	Activate	RBAC: override.activate, Agent guard
POST	/api/overrides/:id/retire	Retire	Tenant
GET	/api/installed-modules/:id/overrides	Get by module	Tenant
GET	/api/installed-modules/:id/resolve	Resolve config	Tenant

RBAC

Method	Path	Purpose	Guards
GET	/api/rbac/permissions	List permissions	None
GET	/api/rbac/roles	List roles	Tenant
POST	/api/rbac/roles	Create role	Admin
POST	/api/rbac/roles/:id/enable	Enable role	Admin
POST	/api/rbac/roles/:id/disable	Disable role	Admin
GET	/api/rbac/roles/:id/permissions	List permissions	Admin
POST	/api/rbac/roles/:id/permissions	Add permission	Admin
DELETE	/api/rbac/roles/:id/permissions/:permissionId	Remove permission	Admin
GET	/api/rbac/users/:userId/roles	List user roles	Admin
POST	/api/rbac/users/:userId/roles	Assign role	Admin
DELETE	/api/rbac/users/:userId/roles/:roleId	Remove role	Admin
GET	/api/rbac/policies	List policies	Admin
POST	/api/rbac/policies	Create policy	Admin
DELETE	/api/rbac/policies/:id	Delete policy	Admin
POST	/api/rbac/seed-defaults	Seed defaults (idempotent)	None
GET	/api/rbac/audit-logs	Get audit logs	Admin

Agent Proposals

Method	Path	Purpose	Guards
GET	/api/agent-proposals	List proposals	Tenant
POST	/api/agent-proposals	Create proposal	Tenant
GET	/api/agent-proposals/:id	Get single	Tenant
POST	/api/agent-proposals/:id/submit	Submit for review	Agent guard (humans only)

POST	/api/agent-proposals/:id/review	Accept/Reject	Agent guard, RBAC: change.approve

Admin Console

Method	Path	Purpose
GET	/api/admin/check-access	Check admin access (RBAC: admin.view)
GET	/api/admin/changes	List all changes (filterable)
GET	/api/admin/approvals	List pending approvals
GET	/api/admin/workflows	List workflow definitions
GET	/api/admin/workflow-executions	List executions
GET	/api/admin/overrides	List overrides
GET	/api/admin/modules	List installed modules
GET	/api/admin/tenants	List tenants
GET	/api/admin/execution-telemetry	Get telemetry events (date filterable)

Admin Graph Introspection

Method	Path	Purpose
GET	/api/admin/graph/snapshot	Get graph snapshot (summary or full)
GET	/api/admin/graph/validate	Validate project graph
GET	/api/admin/graph/diff	Get change diff vs current graph
POST	/api/admin/graph/install	Install graph package (preview or execute)
GET	/api/admin/graph/packages	Get install history
GET	/api/admin/graph/packages/diff	Diff between versions
GET	/api/admin/graph/built-in	List built-in packages
POST	/api/admin/graph/install-built-in	Install built-in package

Admin Environment & Promotion

Method	Path	Purpose	Guards
GET	/api/admin/environments/:envId/packages	Get package state	Admin
GET	/api/admin/environments/diff	Diff packages between environments	Admin
POST	/api/admin/environments/promote	Direct promotion	Admin
GET	/api/admin/environments/promotions	List promotion intents	Admin
POST	/api/admin/environments/promotions	Create intent	Admin
POST	/api/admin/environments/promotions/:id/preview	Preview intent	Admin
POST	/api/admin/environments/promotions/:id/approve	Approve intent	RBAC: environment.promote, Agent guard

POST	/api/admin/environments/promotions/:id/execute	Execute intent	RBAC: environment.promote
POST	/api/admin/environments/promotions/:id/reject	Reject intent	Admin

Vibe Authoring

Method	Path	Purpose
POST	/api/vibe/preview	Generate/refine package and preview
POST	/api/vibe/preview/stream	Stream-based generation with repair (SSE)
POST	/api/vibe/preview/stream-tokens	Token-level streaming (SSE)
POST	/api/vibe/install	Install vibe package
POST	/api/vibe/generate-multi	Generate N variants
POST	/api/vibe/generate-multi/stream	Generate variants with streaming (SSE)

Vibe Drafts

Method	Path	Purpose
GET	/api/vibe/drafts	List drafts
POST	/api/vibe/drafts	Create draft from prompt
POST	/api/vibe/drafts/:draftId/refine	Refine draft
POST	/api/vibe/drafts/:draftId/preview	Preview draft
POST	/api/vibe/drafts/:draftId/install	Install draft (requires previewed)
POST	/api/vibe/drafts/:draftId/discard	Discard draft
POST	/api/vibe/drafts/:draftId/patch	Apply patch ops
GET	/api/vibe/drafts/:draftId/versions	List version history
POST	/api/vibe/drafts/:draftId/restore	Restore to version
POST	/api/vibe/drafts/:draftId/versions/diff	Diff two versions
POST	/api/vibe/drafts/:draftId/adopt-variant	Adopt variant as draft state
POST	/api/vibe/drafts/from-variant	Create draft from variant
POST	/api/vibe/variants/diff	Diff two variant packages

SLA Timers

Method	Path	Purpose
POST	/api/timers/process	Process due SLA timers

Audit

Method	Path	Purpose
GET	/api/audit-feed	Unified audit feed

Record Locks

Method	Path	Purpose
GET	/api/record-locks	List all
GET	/api/record-locks/check	Check if locked

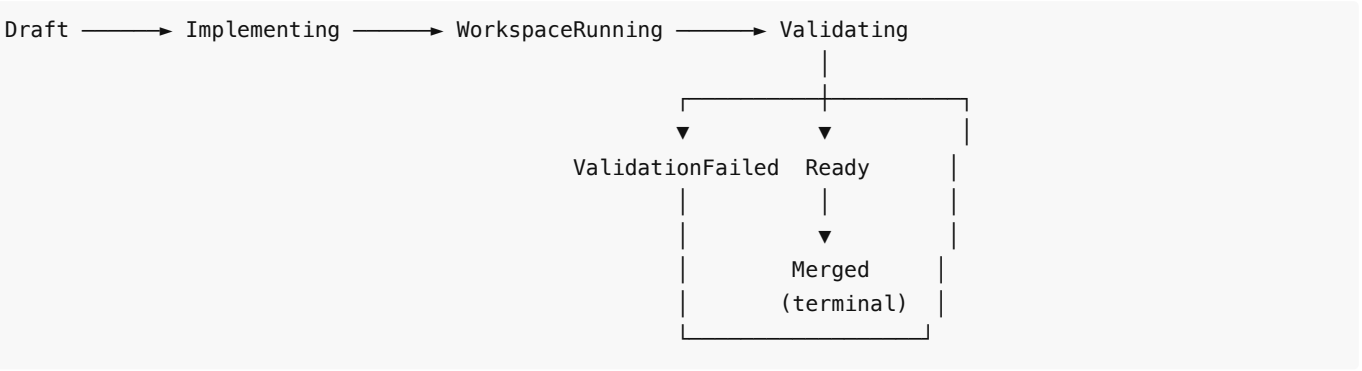
HR Lite

Method	Path	Purpose
POST	/api/hr-lite/install	Install HR Lite template

---

5. State Machines

Change Record Lifecycle



State	Mutable?	Add Targets?	Add Ops?	Execute?
Draft	Yes	Yes	Yes	No
Implementing	Yes	No	Yes	No
WorkspaceRunning	Yes	No	Yes	No
Validating	Yes	No	No	No
ValidationFailed	Yes	No	Yes	No
Ready	Limited	No	No	No
Merged	**No**	**No**	**No**	**N/A**

Patch Op Sub-States

State	executedAt	previousSnapshot	Meaning
Pending	NULL	NULL	Staged, awaiting execution
Executed	Timestamp	Schema JSON	Successfully applied, immutable
Failed	NULL	NULL	Transform rejected, remains pending

Workflow Definition Lifecycle

Workflow Execution Lifecycle

```
Running ──► Completed
   │
   ├──► Failed
   │
   └──► Paused (awaiting approval) ──► Running (on resume)
```

Workflow Execution Intent Lifecycle

```
pending ──► running ──► completed
               │
               └──► failed
pending ──► duplicate (idempotent detection)
pending ──► dispatched (batch processed)
```

Record Type Lifecycle

```
draft ──► active ──► retired
```

Override Lifecycle

```
draft ──► active ──► retired
```

Promotion Intent Lifecycle

```
draft ──► previewed ──► approved ──► executed (terminal)
  │          │              │
  └──────────┴──────────────┴──► rejected (terminal)
```

Vibe Draft Lifecycle

```
draft ──► previewed ──► installed (terminal)
  │          │
  │          └──► draft (on refine/patch/restore)
  │
  └──► discarded (terminal)
```

- Any package mutation (refine, patch, restore, adopt variant) resets status to `draft`
- Install requires `previewed` status
- Terminal states: `installed`, `discarded` — no further mutations

Agent Proposal Lifecycle

```
draft ──► submitted ──► accepted
                  └──► rejected
```

Installed App Status

```
installed ──► upgrading ──► installed
                       └──► failed
```

---

6. 3-Phase Execution Engine

The execution engine (`patchOpExecutor.ts`) applies schema mutations atomically.

Phase 1: Load

1. Verify change exists, has projectId, not Merged, has ops, no ops already executed
2. For each record-type op: load record type, validate target type, cross-project guard (P1)
3. Resolve base-type protected fields
4. Cache: `{ rt, originalSchema, currentSchema, protectedFields }`
5. Emit: `execution_started`

Phase 1.5: Graph Validation (Pre-Mutation)

1. Load ALL tenant record types
2. Simulate pending ops' effect on graph
3. Build projected snapshot
4. Run validators (orphan, cycle, cross-project baseType, duplicate field, binding targets)
5. If errors: emit `graph.validation_failed`, return failure (422)
6. Otherwise: emit `graph.validation_succeeded`, compute diff

Phase 2: Transform (In-Memory, Pure)

For each pending op:

- `applySetField()`: Add/update field, check protected fields
- `applyAddField()`: Add new field, error if exists
- `applyRemoveField()`: Remove field, error if protected or not exists
- `applyRenameField()`: Rename field, error if protected, old not exists, new exists
- **On any error: emit `execution_failed`, return failure (zero DB writes)**

Phase 3: Persist (Only if ALL transforms succeed)

1. For each modified record type:
   - `ensureSnapshot()`: Create snapshot with original schema (idempotent)
   - `updateRecordTypeSchema()`: Write mutated schema
2. Stamp each op with `previousSnapshot` and `executedAt`
3. Emit: `execution_completed`

Key Guarantees

- **All-or-nothing:** Phase 2 failure → zero DB writes
- **Deterministic:** Same inputs → same outputs
- **Snapshots before mutation:** Original state preserved
- **Base-type field protection:** Cannot remove/rename/weaken inherited fields
- **Idempotent snapshots:** Re-execution safe via unique constraint

---

7. Graph Contract Layer

Overview

The graph layer provides a typed overlay on existing data, enabling package management, validation, and cross-environment promotion without new database tables.

Components

Component	Purpose
graphContracts.ts	Pure types: GraphNode, RecordTypeNode, EdgeDefinition, Bindings, GraphSnapshot
graphRegistryService.ts	Builds tenant-scoped graph snapshot from existing tables
graphValidationService.ts	Pure validators (no DB, no throws)
mergeGraphValidator.ts	Phase 1.5 bridge between executor and graph validation

graphDiffService.ts	Deterministic snapshot comparison
installGraphService.ts	Package installation engine (7-phase safety model)
promotionService.ts	Environment-scoped package state and promotion
promotionIntentService.ts	Governed promotion lifecycle
graphService.ts	Admin introspection facade

Graph Validators

Validator	Detects	Error Code
validateNoOrphanRecordTypes	baseType references to non-existent types	ORPHAN_BASE_TYPE
validateNoCyclesInBaseType	Circular inheritance	BASE_TYPE_CYCLE
validateBaseTypeSameProject	Cross-project baseType	BASE_TYPE_CROSS_PROJECT
validateFieldUniquenessPerRecordType	Duplicate field names	DUPLICATE_FIELD
validateBindingTargetsExist	Missing binding targets	BINDING_TARGET_MISSING

Install Engine (7 Phases)

1. **Idempotency:** Same checksum → noop
2. **Version Guard:** Reject downgrade unless allowDowngrade
3. **Ownership Check:** Reject if mutating types owned by other packages
4. **Build Snapshot:** Full tenant snapshot
5. **Project Package:** Simulate package onto snapshot
6. **Validate Projected:** Run all graph validators
7. **Preview or Apply:** If previewOnly → return diff without mutations; else apply topologically (base types first)

Built-in Packages

Package	Key	Contents
HR Lite	hr.lite	employee, department, position record types + workflows
ITSM Lite	itsm.lite	incident, problem, change_request record types + SLA + workflows

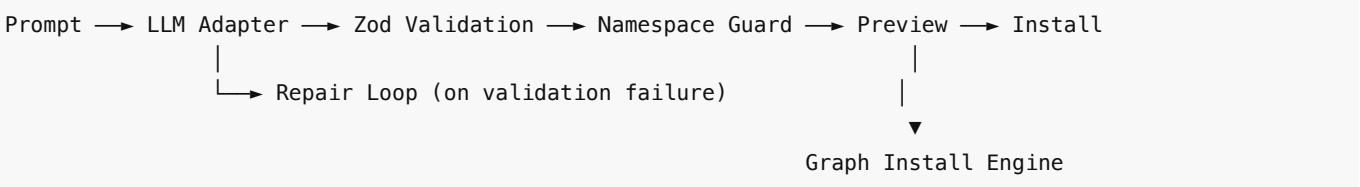
---

8. Vibe Authoring Layer

Overview

AI-powered package generation that creates GraphPackage JSON from natural-language prompts, with validation, repair loops, streaming, and multi-variant exploration.

Flow



LLM Adapters

Adapter	When	Behavior

StubLlmAdapter	Default (no API keys)	Keyword-matched templates, deterministic
AnthropicLlmAdapter	ANTHROPIC_API_KEY set	Claude API, streaming supported
OpenAiLlmAdapter	OPENAI_API_KEY set	GPT-4, streaming supported

Namespace Guards

- Required prefix: `vibe.`
- Blocked prefixes: `hr.` , `itsm.` (reserved for built-in packages)
- Enforced at generation AND validation time (defense-in-depth)

Draft Lifecycle Operations

Operation	Status Change	Creates Version?
Create	→ draft	Yes (reason: "create")
Refine	→ draft	Yes (reason: "refine")
Preview	→ previewed	No
Install	→ installed (terminal)	No
Discard	→ discarded (terminal)	No
Patch Ops	→ draft	Yes (reason: "patch")
Restore Version	→ draft	Yes (reason: "restore")
Adopt Variant	→ draft	Yes (reason: "adopt_variant")

Draft Patch Operations

Op	Parameters	Behavior
add_field	recordTypeKey, field: {name, type}	Adds field (rejects duplicates)
rename_field	recordTypeKey, from, to	Renames (rejects if target exists)
remove_field	recordTypeKey, fieldName	Removes (prevents last field removal)
set_sla	recordTypeKey, durationMinutes	Creates/updates SLA policy
set_assignment_group	recordTypeKey, groupKey	Sets assignment to static_group

Multi-Variant Generation

- Generates 1-5 variants in parallel via `Promise.allSettled`
- Each variant independently validated, namespace-guarded, and diffed
- Variants are ephemeral — never create drafts or mutate graph
- Can adopt a variant into an existing draft or create new draft from variant

Token Streaming

- SSE-based real-time LLM output
- Stages: generation → extract_json → validate_schema → repair → projection → diff → complete
- Preview-only (never installs or creates drafts)

---

9. Workflow & Automation Engine

Step Types

Step Type	Purpose	Config
assignment	Resolve assignee	user/group/rule
approval	Gate with optional auto-approve	required role, auto-approve flag
notification	Emit channel message	channel, message template
decision	Conditional branching	condition, onTrueStepIndex, onFalseStepIndex
record_mutation	Apply data mutations	field updates
record_lock	Lock record	requires record ID from input

Execution Model

- Steps execute in `orderIndex` order
- Decision steps can jump to arbitrary step index (branching)
- Approval steps pause execution (status → awaiting_approval)
- Resume provides outcome → continues from paused step

Trigger System

Trigger Type	Behavior
record_event	Fires on record.created/record.updated, matches record type + field conditions
manual	Explicitly fired via API
schedule	Cron/interval based

Intent Dispatch

- Record events create idempotent intents (deterministic key prevents duplicates)
- `dispatchPendingIntents()` batch-processes all pending
- Race-safe: atomically claims pending → running
- Fire-and-forget: never throws to caller

---

10. RBAC & Agent Guard

Permissions

Permission	Protected Actions
form.view	View forms
form.edit	Edit form overrides
workflow.execute	Manual workflow execution
workflow.approve	Resume paused workflows (approve/reject)
override.activate	Activate module overrides
change.approve	Approve changes (Ready, Merged transitions)
admin.view	Admin console, graph introspection, telemetry
environment.release_create	Create environment releases
environment.promote	Approve/execute promotion intents

Default Roles

Role	Permissions
Admin	All permissions
Editor	form.view, form.edit, workflow.execute, override.activate
Viewer	form.view

Agent Guard

Agents (x-agent-id header) are blocked from:

- Change approval (Ready/Merged transitions)
- Override activation
- Workflow execution/approval
- Trigger firing
- Proposal submission/review
- Promotion intent approval

---

11. Record Instance & SLA System

Creation Flow

1. Validate record type exists and belongs to tenant
2. Resolve assignment via pure `resolveAssignment()` function
3. Create instance with (assignedTo | assignedGroup)
4. If SLA config exists (durationMinutes > 0): create timer, emit `record.sla.created`
5. Emit `record.created` event → triggers matching workflows

Assignment Strategies

Strategy	Config	Behavior
static_user	`{ type: "static_user", value: "user-42" }`	Always assigns to user
static_group	`{ type: "static_group", value: "support-team" }`	Always assigns to group
field_match	`{ type: "field_match", field: "priority", rules: [...], default: {...} }`	Matches record data against rules

SLA Timer Processing

- `processDueTimers(now?, tenantId?)` scans for due timers (status=pending, dueAt <= now)
- Updates each to "breached", emits `record.sla.breached` event
- Idempotent: breached timers not reprocessed
- Fault-tolerant: individual timer errors isolated

---

12. Promotion & Environment Governance

Environments

Each project auto-creates three environments: **dev** (default), **test**, **prod**.

Flag	Purpose
`requiresPromotionApproval`	If true, promotion requires human approval via intent lifecycle
`promotionWebhookUrl`	If set, webhook notifications sent on preview/execute

Promotion Flow

Direct Promotion (no approval):

```
POST /api/admin/environments/promote
  → diffEnvironments(from, to)
  → promoteEnvironmentPackages(from, to)
  → emit graph.package_promoted per package
```

Governed Promotion (with approval):

```
1. Create Intent (draft)
2. Preview Intent → computes diff, sends webhook if configured (draft → previewed)
3. Approve Intent → human actor required (previewed → approved)
4. Execute Intent → applies promotion (approved → executed)
   OR Reject Intent (any → rejected, terminal)
```

Environment Package State

- `environment_package_installs` tracks per-environment package versions
- Diff compares latest checksums between environments
- Packages can be: missing, outdated (checksum differs), or same

---

13. Notification System

Architecture

Best-effort webhook delivery — never blocks promotion lifecycle.

`sendWebhook(url, payload, timeoutMs=5000)`

- Native `fetch` POST with `Content-Type: application/json`
- `AbortSignal.timeout(timeoutMs)` for timeout
- Returns `{ success: true }` on 2xx, `{ success: false, error: string }` otherwise
- **Never throws** — all errors caught and returned as structured result

Webhook Events

Event	Trigger	Payload
`promotion.approval_required`	Preview when requiresPromotionApproval=true AND promotionWebhookUrl set	intentId, projectId, from/to env names, createdBy, diff, timestamp
`promotion.executed`	Execute when promotionWebhookUrl set	intentId, projectId, from/to env names, promoted/skipped counts, timestamp

Notification State Tracking

Tracked on promotion intent: `notificationStatus` (pending/sent/failed), `notificationLastError`, `notificationLastAttemptAt`

---

14. Form System

Compilation Pipeline

1. Load record type, form definition, fields, sections, behavior rules
2. Build CompiledField with effective flags (required/readOnly/visible)
3. Apply behavior rules in orderIndex order
4. Build CompiledSection with CompiledPlacement
5. Apply active form overrides via deep merge

6. Enforce required field invariant

Behavior Rules

Rule Type	Effect
visible	Show/hide field based on condition
required	Make field required based on condition
readOnly	Make field read-only based on condition

Form Patch Operations

- moveField: Reorder field in section
- changeSection: Move field between sections
- toggleRequired/toggleReadOnly/toggleVisible: Set effective flags
- Cannot set required=false on FieldDefinition.isRequired fields

---

15. Service Catalog (All 41 Services)

#	Service	File	Purpose
1	projectService	projectService.ts	Project CRUD, auto-bootstrap
2	changeService	changeService.ts	Change lifecycle, merge flow
3	workspaceService	workspaceService.ts	Dev workspace management
4	agentRunService	agentRunService.ts	Agent execution, skill registry
5	moduleService	moduleService.ts	Module metadata
6	environmentService	environmentService.ts	Environment & release management
7	templateService	templateService.ts	Template catalog (read-only)
8	installService	installService.ts	Template installation
9	overrideService	overrideService.ts	Module config overrides
10	workflowService	workflowService.ts	Workflow definition lifecycle
11	workflowEngine	workflowEngine.ts	Step execution engine
12	triggerService	triggerService.ts	Trigger definition & firing
13	intentDispatcher	intentDispatcher.ts	Async intent dispatch
14	formService	formService.ts	Form definition & compilation
15	rbacService	rbacService.ts	Permissions & roles
16	agentGuardService	agentGuardService.ts	Agent privilege blocking
17	agentProposalService	agentProposalService.ts	Agent-generated proposals
18	auditFeedService	auditFeedService.ts	Unified audit view
19	changeTargetService	changeTargetService.ts	Mutation scope management
20	recordTypeService	recordTypeService.ts	Record type schema
21	patchOpService	patchOpService.ts	Patch op CRUD, guards

22	patchOpExecutor	patchOpExecutor.ts	3-phase execution engine
23	recordInstanceService	recordInstanceService.ts	Runtime record data
24	assignmentService	assignmentService.ts	Pure assignment resolution
25	timerService	timerService.ts	SLA timer processing
26	domainEventService	domainEventService.ts	Event bus (DB + pub-sub)
27	telemetryService	telemetryService.ts	Execution telemetry
28	schedulerService	schedulerService.ts	Background job scheduling
29	hrLiteInstaller	hrLiteInstaller.ts	HR Lite package installer
30	notificationService	notificationService.ts	Webhook delivery
31	graphService	graphService.ts	Admin graph introspection
32	graphRegistryService	graphRegistryService.ts	Graph snapshot builder
33	graphValidationService	graphValidationService.ts	Pure graph validators
34	graphDiffService	graphDiffService.ts	Snapshot diffing
35	installGraphService	installGraphService.ts	Package install engine
36	promotionService	promotionService.ts	Environment promotion
37	promotionIntentService	promotionIntentService.ts	Governed promotion lifecycle
38	vibeService	vibeService.ts	LLM generation & refinement
39	vibeDraftService	vibeDraftService.ts	Draft persistence & versioning
40	multiVariantService	multiVariantService.ts	Multi-variant AI generation
41	tokenStreamService	tokenStreamService.ts	Token-level LLM streaming

Additional supporting modules: variantDiffService, draftVersionDiffService, draftPatchOps, repairService, llmAdapter, graphPackageSchema, vibeTemplates, promptBuilder, mergeGraphValidator, graphContracts.

---

16. Domain Event System

Architecture

Dual-write: DB persistence (`execution_telemetry_events` table) + in-memory pub-sub. Fire-and-forget — never throws, never blocks callers.

Event Structure

```
interface DomainEvent {
  type: DomainEventType;
  status: string;
  entityId: string;
  workflowId?: string | null;
  workflowStepId?: string | null;
  moduleId?: string;     // defaults to "system"
  error?: { code?: string; message: string };
  affectedRecords?: Record<string, unknown> | unknown[] | null;
}
```

Derived Fields

Field	Derivation
actorType	ctx.agentId → "agent", ctx.userId → "user", else "system"
actorId	ctx.agentId ?? ctx.userId ?? null
executionType	workflowId present → "workflow_step", else "task"

Complete Event Type Catalog (50+)

Execution Events (3):

- execution_started, execution_completed, execution_failed

Workflow Events (3):

- workflow.intent.started, workflow.intent.completed, workflow.intent.failed

Record Events (3):

- record.assigned, record.sla.created, record.sla.breached

Graph Events (13):

- graph.validation_failed, graph.validation_succeeded
- graph.diff_computed
- graph.package_installed, graph.package_install_noop, graph.package_install_rejected
- graph.package_promoted
- graph.promotion_intent_created, graph.promotion_intent_previewed, graph.promotion_intent_approved, graph.promotion_intent_executed, graph.promotion_intent_rejected
- graph.promotion_notification_sent, graph.promotion_notification_failed

Vibe Events (25+):

- vibe.llm_generation_requested, vibe.llm_generation_succeeded, vibe.llm_generation_failed
- vibe.llm_refinement_requested, vibe.llm_refinement_succeeded, vibe.llm_refinement_failed
- vibe.llm_repair_attempted
- vibe.draft_created, vibe.draft_patched, vibe.draft_refined, vibe.draft_previewed, vibe.draft_installed, vibe.draft_discarded, vibe.draft_restored, vibe.draft_variant_adopted
- vibe.variant_generation_requested, vibe.variant_generation_completed
- vibe.variant_diff_computed
- vibe.llm_token_stream_started, vibe.llm_token_stream_completed, vibe.llm_token_stream_failed
- vibe.draft_version_created, vibe.draft_version_diff_computed

In-Memory Pub-Sub

```
const unsubscribe = subscribe("record.sla.breached", (ctx, event) => { ... });
unsubscribe();
clearSubscribers(); // test isolation
```

---

17. Current Client Architecture

Tech Stack

- **React 19** with TypeScript
- **Vite** for bundling and HMR
- **TanStack Query** for data fetching (staleTime: Infinity, no auto-refetch)
- **Wouter** for client-side routing
- **Tailwind CSS** + **shadcn/ui** component library

- **localStorage** for tenant/user identity (not sessions)

Client-Side Patterns

1. **No sessions/cookies:** Tenant and user IDs sent as headers on every request
2. **Infinite stale time:** Data never auto-refetches; invalidated explicitly on mutation
3. **No auto-retry:** Failures surfaced immediately
4. **Path-based query keys:** Mirror API routes for intuitive cache management
5. **SSE streaming:** Token-level streaming for Vibe generation
6. **Mutation → invalidate pattern:** All mutations followed by queryClient.invalidateQueries()

Tenant Bootstrap

- On first load: fetches `/api/tenants`, auto-selects "default" or first available
- Validates slug format (rejects stale UUIDs)
- Supports manual tenant switching

---

18. Current UI Pages & Components

Page Inventory (13 Routes)

Route	Page	Description
`/`	Dashboard	Landing page
`/projects`	Projects	Project list
`/projects/:id`	Project Detail	Single project view
`/changes`	Changes	Change list
`/changes/:id`	Change Detail	Single change view
`/skills`	Agent Skills	Skill registry
`/runner`	Runner	Terminal interface
`/studio/forms`	Form Studio	Form editing
`/admin/:section?`	Admin Console	Multi-tab admin (6 panels)
`/workflow-monitor`	Workflow Monitor	Execution monitoring
`/records`	Records	Record instances with SLA
`/vibe-studio`	Vibe Studio	AI package generation
`*`	NotFound	404 page

Navigation Structure (AppSidebar)

Navigation Section:

- Dashboard (home)
- Projects (folder)
- Changes (git pull request)

Tools Section:

- Agent Skills (bot)
- Runner (terminal)
- Form Studio (pen tool)
- Vibe Studio (sparkles)
- Admin (shield)

- Workflow Monitor (activity)
- Records (database)

Admin Console (6 Panels)

Panel	Features
TenantsPanel	Tenant list, click-to-switch, active highlight
AppsPanel	Installed modules with version/status
OverridesPanel	Expandable override rows, activate button
ApprovalsPanel	Pending approvals with approve/reject buttons
WorkflowsPanel	Definitions + recent executions
ChangesPanel	Filterable by status/date, status badges

Vibe Studio (3 Main Areas)

Left Panel (Draft Management):

- Creation prompt textarea
- Optional app name input
- Create Draft / Compare Variants / Token stream toggle buttons
- Draft list with status badges

Main Content - Drafts Tab:

- Variant comparison grid (when variants exist)
- Selected draft: header, meta info, action buttons (Refine, Preview, Install, Discard)
- Token streaming output panel
- Preview diff viewer
- Package contents table
- Version history with restore
- Inline edit section (add_field, rename_field, remove_field, set_sla, set_assignment_group)

Main Content - Promotion Tab:

- Environment from/to selectors
- Package state tables (side-by-side)
- Drift analysis (compute drift, promote)
- Promotion intent management (create, preview, approve, execute, reject)

Records Page

- Record type selector dropdown
- Process Timers button
- Instance table: ID, Assigned, SLA Due, SLA Status, Created By, Created
- Detail panel for expanded view
- SLA status badges (Breached=red, Pending=green)

---

19. Client API Layer

Core API Helper (`lib/queryClient.ts`)

```
// Tenant/user management
getTenantId() / setTenantId(id)
getUserId() / setUserId(id)

// Request helpers
```

```
tenantHeaders() → { "x-tenant-id": ..., "x-user-id": ... }
apiRequest(method, url, data?) → Response
getQueryFn(options?) → TanStack query function
```

Vibe API Client (`lib/api/vibe.ts`)

Function	Parameters	Returns
createDraft	projectId, prompt, appName	VibeDraft
listDrafts	projectId?	VibeDraft[]
refineDraft	draftId, prompt	VibeDraft
previewDraft	draftId	VibeDraft
installDraft	draftId	{ draft, installResult }
discardDraft	draftId	VibeDraft
patchDraft	draftId, ops	VibeDraft
listDraftVersions	draftId	DraftVersion[]
restoreDraftVersion	draftId, versionNumber	VibeDraft
generateMulti	projectId, prompt, count, appName	{ variants }
createDraftFromVariant	projectId, prompt, package	VibeDraft
diffVariants	projectId, packageA, packageB	VariantDiffResult
adoptVariantIntoDraft	draftId, package, prompt	VibeDraft
diffDraftVersions	draftId, fromVersion, toVersion	VersionDiffResult
streamPreview	body, onStage	StreamResult (SSE)
streamPreviewTokens	body, onEvent	TokenStreamResult (SSE)

Promotion API Client (`lib/api/promotion.ts`)

Function	Parameters	Returns
createPromotionIntent	projectId, fromEnvId, toEnvId	PromotionIntent
previewPromotionIntent	intentId	PromotionIntent
approvePromotionIntent	intentId	PromotionIntent
executePromotionIntent	intentId	PromotionIntent
rejectPromotionIntent	intentId	PromotionIntent
listEnvironments	projectId	EnvironmentInfo[]
listEnvironmentPackages	envId	EnvironmentPackageState[]
diffEnvironments	fromEnvId, toEnvId	EnvironmentDiffResult
listPromotionIntents	projectId	PromotionIntent[]

---

20. System Invariants (Complete)

Data Invariants (D1-D9)

- **D1:** Record types have non-null project_id
- **D2:** Record type keys unique per (tenant_id, project_id, key)
- **D3:** Record type names unique per tenant
- **D4:** Changes have non-null project_id
- **D5:** Snapshots unique per (change_id, record_type_key) — idempotent
- **D6:** Snapshots persist change's project_id, not record type's
- **D7:** Op previous_snapshot + executed_at set atomically
- **D8:** Field types constrained to: string, number, boolean, reference, choice, text, date, datetime
- **D9:** Change targets have typed selectors matching target type

Execution Invariants (E1-E7)

- **E1:** Merged changes immutable
- **E2:** Change cannot be executed twice (rejects if any op has executedAt)
- **E3:** Executed ops cannot be deleted (409 Conflict)
- **E4:** Execution all-or-nothing (Phase 2 failure → zero DB writes)
- **E5:** Execution deterministic (same inputs → same outputs)
- **E6:** Snapshots captured before mutation
- **E7:** Base-type protected fields cannot be weakened/removed/renamed

Project Invariants (P1-P7)

- **P1:** Patch ops target record types in same project as change
- **P2:** Snapshots inherit project_id from change
- **P3:** Record type creation requires existing project
- **P4:** Changes list by project
- **P5:** Change targets inherit project_id from parent change
- **P6:** Base types must belong to same project as derived record type
- **P7:** Patch op creation validates project consistency early (fail-fast)

Tenant Invariants (T1-T6)

- **T1:** Tenant ID resolved server-side from slug
- **T2:** All data access tenant-scoped via WHERE clause
- **T3:** Missing tenant header → 401
- **T4:** Unknown slug → 404
- **T5:** Op deletion validates tenant ownership
- **T6:** No sessions, cookies, or body-supplied tenant IDs

RBAC Invariants (R1-R3)

- **R1:** Agents cannot perform approval/privileged actions (403)
- **R2:** RBAC per-tenant
- **R3:** System actors bypass RBAC

Operational Invariants (O1-O95)

- **O1-O2:** Domain events never throw, subscriber errors isolated
- **O3-O6:** SLA processing idempotent, assignment pure, record event intents idempotent
- **O7-O12:** Graph validation before mutation, diff computation, merge boundary
- **O13-O29:** Install engine (topological order, checksum idempotency, version guard, ownership, environment state)
- **O30-O36:** Promotion governance (state machine, environment gates, human approval, intent audit)
- **O37-O48:** Vibe layer (LLM untrusted, Zod validation, namespace guard, repair loop, refinement fallback)
- **O49-O66:** Draft persistence, versioning, patching, multi-variant (max 5, ephemeral)
- **O67-O76:** Token streaming (preview-only), variant comparison (pure), draft version diff (read-only)
- **O77-O91:** Additional vibe/graph operational guarantees
- **O92-O95:** Notification system (best-effort, never blocks lifecycle, tracked on intent, observable via telemetry)

---

21. Phase History (26 Phases)

Phase	Date	Summary
0-4	2025-01-01 — 2025-01-20	Core primitives, invariants, state machine, API contracts
5-5.2	2025-05-19 — 2025-05-20	Graph contract layer (validation, diff, preview)
6-7	2025-05-20 — 2025-06-15	Graph install engine (package install, versioning, audit)
8-9	2025-06-15 — 2025-07-01	Built-in packages (HR/ITSM Lite), bindings support
10-12	2025-07-01 — 2025-09-01	Promotion system (cross-environment, intents, gating)
13-15	2025-09-01 — 2025-10-01	Vibe authoring (LLM generation, repair, streaming)
16-17	2025-10-01 — 2025-11-01	Vibe UI, draft persistence
18-19	2025-11-01 — 2025-12-01	Refinement, token streaming
20-21	2025-12-01 — 2026-01-01	Inline editing (patch ops, versioning, restore)
22-23	2026-01-01 — 2026-01-15	Multi-variant AI (generation, diff)
24-25	2026-01-15 — 2026-02-15	Version-to-version diff, token streaming
26	**2026-02-21**	**Promotion notifications (webhook approval/execution)**

Current Stats

- **656 tests** passing across 43 test files
- **~50 database tables**
- **~150 API endpoints**
- **41 services**
- **50+ domain event types**
- **95 operational invariants**

---

22. UI/UX Design Considerations

What Exists Today

The current UI is functional but minimal — built to validate platform capabilities rather than deliver a polished user experience. Key gaps:

Pages with Substantial UI

1. **Admin Console** — 6 panels (tenants, apps, overrides, approvals, workflows, changes)
2. **Vibe Studio** — Full AI generation workflow + promotion management
3. **Records** — Instance browser with SLA tracking

Pages Needing UI/UX Design

1. **Dashboard** — Currently minimal, needs metrics/status overview
2. **Projects** — List view, needs detail pages with environment/module management
3. **Project Detail** — Needs changes, record types, environments, modules tabs
4. **Changes** — List view, needs detail with target/op/execution views
5. **Change Detail** — Needs lifecycle visualization, op management, diff viewer
6. **Form Studio** — Needs full form builder with section/field/rule management
7. **Workflow Monitor** — Needs execution timeline, step-by-step viewer, approval UI
8. **Agent Skills** — Needs skill catalog, run history

Key Data Relationships for UI

```
Tenant
  └── Projects
       ├── Environments (dev, test, prod)
       │    └── Package Installs
       ├── Modules
       ├── Record Types
       │    ├── Fields
       │    ├── Instances
       │    │    └── Timers (SLA)
       │    └── Forms
       │         ├── Sections → Placements
       │         └── Behavior Rules
       ├── Changes
       │    ├── Targets
       │    ├── Patch Ops → Snapshots
       │    └── Workspaces / Agent Runs
       └── Workflow Definitions
            ├── Steps
            ├── Triggers
            └── Executions
                 └── Step Executions
```

User Personas (Implied by RBAC)

Persona	Role	Key Actions
Admin	Admin	Full platform control, graph management, telemetry, promotion approval
Editor	Editor	Form editing, workflow execution, override activation
Viewer	Viewer	Read-only form access
Agent	Agent	Automated actions (blocked from approval/privileged actions)

Critical UI Workflows to Design

1. **Change Lifecycle:** Draft → add targets → add ops → execute (test) → approve → merge
    - Needs: status timeline, target/op management panels, diff viewer, approval actions
2. **Vibe Authoring Flow:** Prompt → generate variants → compare → select/adopt → refine → preview → install
    - Exists but could be improved with better variant comparison, inline editing UX
3. **Promotion Governance:** Select environments → compute drift → create intent → preview → approve → execute
    - Exists in Vibe Studio Promotion tab, may warrant dedicated page
4. **Record Instance Management:** Select type → view instances → SLA tracking → assignment routing
    - Basic exists, needs detail view, SLA dashboard, assignment override
5. **Workflow Builder:** Create definition → add steps (assignment, approval, notification, decision) → activate → monitor executions
    - Needs visual step builder, execution timeline, approval queue
6. **Form Builder:** Select record type → manage sections → place fields → add behavior rules → apply overrides
    - Needs drag-and-drop builder, behavior rule editor, live preview

Available API Surface Not Yet in UI

Feature	Endpoints Available	UI Status
Release management	POST /environments/:id/release	No UI
Graph introspection	GET /admin/graph/snapshot, validate, diff	Admin panel only
Package install history	GET /admin/graph/packages	No dedicated UI
Audit feed	GET /audit-feed	No UI
Record locks	GET /record-locks	No UI
RBAC management	Full RBAC CRUD endpoints	No UI
Agent proposals	Full CRUD + submit/review	No UI
Template catalog	GET /templates	No UI (install only)
Telemetry events	GET /admin/execution-telemetry	Admin panel only
Choice list management	Full CRUD	No UI
Workflow trigger management	Full CRUD	Limited UI

Design Constraints

- **Stateless:** No sessions — all identity from headers
- **Multi-tenant:** Tenant switcher, all data scoped
- **RBAC-aware:** Hide/disable actions based on permissions
- **Agent-aware:** Distinguish human vs agent actors
- **Streaming:** SSE for real-time LLM output
- **Best-effort notifications:** Webhook status may lag

End of Architecture Reference — 656 tests, 26 phases, 50+ tables, 150+ endpoints, 41 services, 95 invariants.