SkillPilot Developer Handbook · Volume VII

Developer, Schema, and Editor Guide

Documentation Guidelines, Landscape JSON Field Reference, Runtime Fallbacks,

Authoring Editors, Sibling Node Ordering, Liquibase, and SSE Auto-Update

Scope of this guide

This volume consolidates SkillPilot's durable developer, schema, authoring-tool, and platform references: documentation guidelines including all eleven bilingual single-source compendia and the MkDocs `[ DE | EN ]` switcher, the `SkillLandscape` JSON field reference, minimal projection schema and runtime fallbacks, the four local Vite curriculum editors, balanced topological sibling ordering in 'My Goals', database schema migrations via Liquibase, and real-time Server-Sent Events (SSE) auto-updates.

As of 11 November 2026 · License: Apache-2.0 (software, schema, and developer documentation; see LICENSING.md).

Dated migration workboards and rollout dossiers under docs/dev/ remain standalone working records.

Contents

1 Documentation Guidelines **3**
1.1 Document Roles and Placement Rules . . . 3
1.2 Bilingual Single-Source Compendia (`docs/compendia.json`) . . . 3
1.3 MkDocs `[ DE | EN ]` Switcher, Generated Notices, Templates, and Checks . . . 4

2 Landscape JSON Field Reference **5**
2.1 Top-Level `SkillLandscape` Fields . . . 5
2.2 `LearningGoal`, `dimensionTags`, and Provenance Fields . . . 5

3 Minimal Projection Schema, Runtime Fallback, and Math Pilot **6**
3.1 Additive Schema Fields and Fallback Precedence . . . 6
3.2 Entry-Scope Resolution and Canonical Mathematics Pilot . . . 6

4 Curriculum Authoring Editors **7**
4.1 Four Local Vite Editors and Route Architecture . . . 7
4.2 Three-Pane Layout, Blocking vs. Warning Validations, and Rollout Order . . . 7

5 Graph Editor **8**
5.1 Focused `requires` Refactoring Workflow . . . 8

6 Flashcard Editor **9**
6.1 Deck JSON Schema, Bilingual Pairing, and Dual-Write Save . . . 9

7 Sibling Node Ordering in 'My Goals' **10**
7.1 Hard Direct `requires` Topological Sort Plus Six-Tier Tie-Breaker . . . 10

8 Database Migrations with Liquibase **11**
8.1 Changelog Layout and Four-Step Migration Workflow . . . 11

9 Real-Time Auto-Update Architecture (SSE) **12**
9.1 Backend Event Bus, SSE Endpoint, and React Reconnection Hook . . . 12

1 Documentation Guidelines

Every documentation file in `docs/` has one primary role so durable concepts, QA rules, generated status snapshots, and dated dossiers stay cleanly separated.

1.1 Document Roles and Placement Rules

| Role | Purpose | Canonical Location | Source of Truth? | | — | — | — | — | | **Concept** | Durable domain model, semantics, and architecture | `docs/concept/` | Yes | | **Runbook** | Repeatable operational command steps | Near the owning process doc | Yes | | **Process** | Review rules, status meanings, and decision gates | `docs/qa-ci/` or `docs/dev/` | Yes | | **Generated status** | Computed report, queue, or dashboard snapshot | `docs/qa-ci/status/` | No | | **Pilot or dossier** | Dated investigation, PoC, or one-off audit record | `docs/qa-ci/archive/` or `docs/dev/` | No (unless stated) | | **Implementation note** | Developer-facing schema, tool, or migration detail | `docs/dev/` (`tmp/` for throwaway work) | Sometimes | {cols="20%,34%,28%,18%"}

1.2 Bilingual Single-Source Compendia (`docs/compendia.json`)

Durable reference handbooks are authored as bilingual (`de`/`en`) single-source compendia compiled by `scripts/doc_compendium/` and `build_compendia.py` into MkDocs Web Markdown pages and typeset LuaLaTeX PDFs. Never mix languages in one file and never hand-edit generated Web pages; edit `source/chapters/*.{de,en}.md` with 1:1 AST block parity, identical `{#sec:...}` / `anchors="..."`, `@ref{sec:...}`, canonical `repo:...` links, and pure ASCII code blocks (`<=` 72 chars/line). Contract-pinned deployment/freeze records, dated workboards, and generated status reports remain standalone Markdown files.

| Compendium ID | Canonical `source/` | Tracking Policy | Generated PDFs (`DE` / `EN`) | | — | — | — | — | | `skill-graph` | `docs/concept/skill-graph/source/` | `untracked` | `skillpilot-graph-definition.{pdf,en.pdf}` | | `skill-graph-architecture` | `docs/concept/skill-graph/architecture-source/` | `committed-web` | `skillpilot-skill-graph-architecture.{de,en}.pdf` | | `whitepaper` | `docs/whitepaper/source/` | `committed-web` (+ DE PDF) | `whitepaper.{de,en}.pdf` | | `glossary` | `docs/concept/glossary/source/` | `committed-web` | `skillpilot-glossary.{de,en}.pdf` | | `didactic` | `docs/concept/didactic/source/` | `committed-web` | `skillpilot-didactics-and-planning.{de,en}.pdf` | | `runtime-workflows` | `docs/concept/runtime-workflows/source/` | `committed-web` | `skillpilot-coach-and-runtime-architecture.{de,en}.pdf` | | `production-pipelines` | `docs/production-pipelines/source/` | `committed-web` | `skillpilot-production-pipelines.{de,en}.pdf` | | `qa-ci` | `docs/qa-ci/source/` | `committed-web` | `skillpilot-curriculum-quality-handbook.{de,en}.pdf` | | `qa-ci-lanes` | `docs/qa-ci/lanes-source/` | `committed-web` | `skillpilot-qa-lanes-and-release-contracts.{de,en}.pdf` | | `security` | `docs/security/source/` | `committed-web` | `skillpilot-security-and-privacy.{de,en}.pdf` | | `dev-handbook` | `docs/dev/source/` | `committed-web` | `skillpilot-developer-and-editor-guide.{de,en}.pdf` | {cols="22%,32%,18%,28%" longtable=true}

```
python3 scripts/build_compendia.py --web-and-tex
python3 scripts/build_compendia.py --pdfs
python3 scripts/build_compendia.py --check-committed-web
python3 scripts/build_compendia.py --check-untracked-outputs
```

1.3 MkDocs [DE | EN] Switcher, Generated Notices, Templates, and Checks

- **Site navigation & [DE | EN] pill switcher:** mkdocs.yml (site_url: https://enpasos.github.io/skillpilot/) pairs Compendium (EN) / Kompendium (DE) nav groups and registers standalone pairs under standaloneBilingualPairs in docs/compendia.json. language-switcher.js + bilingual-map.js toggle active-language pages while keeping monolingual operational/dated pages visible in both modes. Unlisted dated dossiers and docs/qa-ci/status/ reports remain reachable via section index.md files.
- **Generated notices & templates:** Generated Markdown files declare Generated by, Regenerate with, and Source of truth headers registered in generatedMarkdownNoticeRegistry.ts (npm run check:generated-doc-notices, npm run check:generated-status-registry). Durable QA lane docs follow Purpose / Source Of Truth / Generated Artifacts / Run / Blocking Semantics / Review Ledger / Review Workflow / Expansion Criteria / Related Documents; runbooks follow Run The Audit / Inspect Results / Decide Or Fix An Item / Interpret Diagnostics / Verify Local Changes; PoC/archive records carry ISO dates in filenames.
- **Terminology, outside-docs/ links, and CI checks:** Core vocabulary follows the Glossary and checkTerminology.ts (npm run check:terminology). Links leaving docs/ must use GitHub https://github.com/enpasos/skillpilot/blob/main/... (or /tree/main/...) URLs so published site pages never 404, verified by .github/workflows/docs_checks.yml:

```
git diff --check
cd app
npm run check:terminology
npm run check:docs-links
npm run check:docs-indexes
```

2 Landscape JSON Field Reference

This chapter documents the field-level authoring and runtime contract for `SkillLandscape` JSON files validated against `docs/landscape-runtime.schema.json` and mapped into `UiGoal` in `goalTypes.ts` (see node semantics in Node Types and projection contracts in View Projection).

2.1 Top-Level `SkillLandscape` Fields

| Field | Required? | Purpose & Constraints | | — | — | — | | `landscapeId` | Yes | Globally unique stable identifier of the skill landscape. | | `locale` | Yes | Primary authoring locale (`"de"` or `"en"`). | | `subject` | Yes | Canonical subject key used for routing, grouping, and asset paths. | | `title`, `description` | Yes | Localized title and overview (`titleEn`, `descriptionEn` optional for inline EN translations). | | `frameworkId`, `filters` | Optional | Institutional framework tag and selectable scope filter definitions (for example `GK`, `LK`, jurisdiction). | | `programUnits`, `goalPlacements`, `competencyCatalog` | Optional | Additive projection-layer arrays (3). | | `goals` | Yes | Ordered array of `LearningGoal` records forming the `contains` and `requires` DAGs. | {cols="26%,14%,60%"}

2.2 `LearningGoal`, `dimensionTags`, and Provenance Fields

| Field Group | Fields & Runtime Interpretation | | — | — | | **Identity & text** | `id` (stable UUID, never reused for new meaning), optional landscape-unique `shortKey`, `title` (`titleEn`), `description` (`descriptionEn`). | | **Graph relations** | `contains` (ordered child goal IDs; empty `[]` = atomic goal, non-empty = cluster goal) and `requires` (prerequisite goal IDs; target state strictly atomic-to-atomic). | | **Classification & weight** | `weight` (numeric workload/progress weight), optional `core` (`boolean`), `courseLevel`, `tags` (`string[]`), `semanticKind` / `nodeKind` / `type`, `applicability`, and `competencyRefs`. | | **Pedagogical payloads** | `examples`, `extendedData` (including `applicabilityOverrides` and manual tree-order hints), `examData` (`taskContent`, `solutionContent`, `scoring`), and `experimentData`. | | **`dimensionTags`** | Structured metadata object: `phase` (`E`, `Q1`–`Q4`, `SekI`, `GLOBAL`, compatibility/badge field), `demandLevel` (`AB1`–`AB3`), `processCompetencies`, `guidingIdeas`, `framework`, `area`, `topicCode`. | | **Provenance & media** | `sourceRef` (official curriculum locator) and `resourceLinks` (`type: "curriculum"` or `"goal-visualization"`, never new external provider links on canonical goals; see Source & Resource Links). | {cols="26%,74%"}

At runtime, frontend loaders normalize `LearningGoal` entries into `UiGoal` (`goalTypes.ts`) and evaluate scope filters, frontier availability, SRS decks, and mastery aggregation without mutating authored JSON identities.

3 Minimal Projection Schema, Runtime Fallback, and Math Pilot

To evolve existing landscapes into the layered architecture (General Goal System) without breaking unmigrated curricula, SkillPilot introduces four optional additive JSON structures paired with deterministic runtime fallback rules.

3.1 Additive Schema Fields and Fallback Precedence

```
{
  "programUnits": [
    { "id": "sek2-e", "kind": "phase", "label": "E-Phase", "order": 10 }
  ],
  "goalPlacements": [
    {
      "goalId": "<uuid>",
      "unitId": "sek2-e",
      "relation": "primary",
      "context": { "jurisdiction": "DE-HE", "stage": "SekII" }
    }
  ],
  "competencyCatalog": [
    { "id": "K1", "label": "Mathematisch argumentieren", "dimension": "process" }
  ]
}
```

Goals additionally carry optional `competencyRefs: ["K1", ...]`. Runtime precedence follows strict non-destructive rules:

1. **Composition view first:** If a reviewed `.view.json` matches the resolved learner scope, compile the default tree from that composition view plus canonical `contains`.
2. **Explicit placement projection second:** Otherwise, when `programUnits` and matching `primary goalPlacements` exist, project the program tree directly from those arrays.
3. **Legacy `contains` / `phase` fallback third:** Landscapes without placement or view artifacts continue rendering their authored `contains` hierarchy and `dimensionTags.phase` badges unchanged.
4. **Unchanged mastery & frontier:** Mastery storage and `requires` frontier calculation always operate on canonical atomic `LearningGoal` IDs, never on `programUnits` or view structure nodes.

3.2 Entry-Scope Resolution and Canonical Mathematics Pilot

Entry-scope resolution executes in five deterministic steps: (1) select base root curriculum, (2) bind jurisdiction / duration / stage / course profile (`Level 2`), (3) resolve matching `.view.json` or placement context, (4) close required atomic prerequisites $L = \text{Closure}(S)$ on the unpruned canonical graph, and (5) apply optional `Level 3` focus filters. `curricula/DE/Gymnasium/canonical/DE_DEU_S_GYM_CANONICAL_MATHEMATIK.de.json` serves as the reference pilot proving additive `programUnits`, `goalPlacements`, `competencyCatalog`, and state-scoped `.view.json` compilation side by side.

4 Curriculum Authoring Editors

SkillPilot separates local Vite development-server authoring tools by semantic layer rather than mixing canonical graph editing, flashcard curation, and scope-view composition into one monolithic UI.

4.1 Four Local Vite Editors and Route Architecture

| Editor | Frontend Route & Dev Middleware | Owned Artifact Layer | | — | — | — | | **Flashcard Editor** (6) | `/flashcard-editor` · `/__deck-editor/*` | `*_deck*.json` flashcard decks (`curricula/...` + `app/public/data/`). | | **Graph Editor** (5) | `/graph-editor` · `/__graph-editor/*` | Narrow `requires` refactoring and cluster-to-atomic prerequisite normalization on landscape JSONs. | | **Canonical Cluster Editor** | `/canonical-cluster-editor` · `/__canonical-cluster-editor/*` | Canonical `contains` cluster hierarchy, sibling order, and cluster metadata in `curricula/DE/Gymnasium/canonical/`. | | **Composition View Editor** | `/composition-view-editor` · `/__composition-view-editor/*` | Scope-specific `*.view.json` upper trees (`structure` nodes + cluster-only `canonicalSubtree` attachments). | {cols="26%,34%,40%"}

All `/__*` file-mutation endpoints exist **exclusively** inside local Vite dev-server middleware (`vite.config.ts`) and are never exposed in production builds.

4.2 Three-Pane Layout, Blocking vs. Warning Validations, and Rollout Order

Both V1 structural editors use a three-pane layout (left tree navigator, center node/attachment inspector, right live compiled-preview & diagnostics pane):

- **Blocking save validations (reject write):** Malformed JSON, duplicate IDs, `contains` cycles, dangling `goalId` references, authoring atomic goals inline inside `.view.json`, or overlapping `canonicalSubtree` expansions violating single-occurrence projection (View Projection).
- **Non-blocking warnings (allow save with diagnostic badge):** Unreferenced canonical clusters in a partial draft view, `unplacedInScope` candidates, or pending downstream QA reruns (`npm run validate:graph`, `validate:views`).
- **Four-phase implementation sequence:** (1) Read-only compiled tree & diagnostic inspector → (2) `Canonical Cluster Editor` save pipeline → (3) `Composition View Editor .view.json` authoring → (4) cross-view coverage diffing (deliberate non-goals: multi-user cloud editing, automatic AI graph rewriting, or editing learner state).

5 Graph Editor

The local Graph Editor (`/graph-editor`, implemented in `GraphEditorView.tsx` backed by Vite dev middleware `/__graph-editor/{list,load,save}`) is a focused refactoring workbench for `requires` prerequisites in `SkillLandscape` JSON files.

5.1 Focused `requires` Refactoring Workflow

Rather than acting as a free-form canvas, the editor targets canonical `requires` hygiene ($R \subseteq A \times A$; see Graph Definition):

- **Split prerequisite inspection:** Separates every selected goal's `requires` edges into **atomic prerequisites**, **legacy cluster prerequisites**, and **unresolved/external IDs** so legacy cluster links stand out immediately.
- **Guided cluster-to-atomic expansion:** Inspects the atomic descendants of a required cluster so authors can select the exact required atomic competencies before removing the legacy cluster edge and running transitive reduction.
- **In-place 2-space JSON persistence:** `/__graph-editor/save` writes formatted 2-space UTF-8 JSON directly back to the source file under `curricula/` without mirroring to `app/public/data/`.

```
cd app
npm run dev
# After saving changes in http://localhost:5173/graph-editor:
npm run validate:graph
```

6 Flashcard Editor

The local Flashcard Editor (`/flashcard-editor`, implemented in `FlashcardEditorView.tsx` and `FlashcardFlipCard.tsx` backed by Vite dev middleware `/__deck-editor/{list,load,save}`) authors and previews spaced-repetition decks (`*_deck*.json`) with live KaTeX math and Markdown rendering.

6.1 Deck JSON Schema, Bilingual Pairing, and Dual-Write Save

```
{
  "deckId": "physics_sek2_q1_fields_deck",
  "title": "Q1 Elektrisches und magnetisches Feld",
  "cards": [
    {
      "id": "q1-field-01",
      "front": "Definiere die elektrische Feldstaerke $E$.",
      "back": "$E = \\frac{F}{q}$ in $\\mathrm{N/C}$ bzw. $\\mathrm{V/m}$.",
      "category": "Grundbegriffe",
      "tags": ["GK", "LK"]
    }
  ]
}
```

Authoring invariants:

- **Required card fields:** Every deck defines `deckId`, `title`, and `cards[]`; each card carries a deck-unique `id`, `front`, `back`, `category`, and optional `tags` (`string[]`). Card IDs must remain stable so existing learner SM-2 and `verifiedRecall` records (Node Types) stay bound.
- **DE / EN deck pairing:** Optional English companion decks (`*_deck.en.json`) align card-by-card via identical `id` values so learners can switch languages without losing recall state.
- **Dual-write persistence (repository mode):** Saving via `/__deck-editor/save` writes both the canonical file under `curricula/...` and its runtime mirror under `app/public/data/<basename>.json` (whereas standalone package mode resolves decks from `data/cards/` via `CurriculumResourceResolver`).

7 Sibling Node Ordering in 'My Goals'

In the learner-facing competence tree (`CompetenceTree.tsx`), sibling nodes under a cluster are ordered deterministically by `sortSiblingGoalsBalanced` in `goalSorter.ts` so prerequisites appear before dependents while preserving intentional didactic structure.

7.1 Hard Direct `requires` Topological Sort Plus Six-Tier Tie-Breaker

Given sibling goals under one parent cluster (guaranteed acyclic on `requires`), `sortSibling GoalsBalanced` executes Kahn's topological sort over **direct sibling-to-sibling `requires` edges** and resolves ready-queue ties strictly in six lexicographic tiers:

| Priority | Criterion | Exact Rule | | — | — | — | | **1. Manual order** | Explicit authoring override | Numeric `extendedData.{treeOrder,sortOrder,displayOrder,order,position}` or tag `order:<n>`. | | **2. Didactic bucket** | Pedagogical framing | `why` (orientation/motivation) < `tutorial` / core competencies < `flashcards` (memory) < `exercises` / exams. | | **3. Phase order** | Curricular progression | `SekI` / `5–10` < `E` < `Q1` < `Q2` < `Q3` < `Q4` < `Abitur` / `GLOBAL`. | | **4. Descendant support** | Soft cross-cluster `requires` | `softScore(v)` over descendant-level prerequisite dependencies gated by `minAbsoluteSupport = 2` and `minRelativeSupport = 0.25` so isolated single edges do not unexpectedly reorder large clusters. | | **5. Localized title** | Human-readable fallback | Collator `localeCompare` (`de-DE`, `numeric: true`). | | **6. Canonical ID** | Final determinism | Lexicographic `goal.id` comparison ensuring 100% deterministic output. | {cols="20%,24%,56%"}

Note that default raw content-tree inspection (View Projection) preserves authored `contains` / `treeOrder` sequence, whereas `sortSiblingGoalsBalanced` powers the didactic learner navigation view when explicit `treeOrder` overrides are absent.

8 Database Migrations with Liquibase

The Spring Boot backend enforces `spring.jpa.hibernate.ddl-auto=validate`: Hibernate validates JPA entities against the live schema on startup, while **Liquibase** exclusively owns every schema creation and migration.

8.1 Changelog Layout and Four-Step Migration Workflow

- **Master changelog:** `backend/src/main/resources/db/changelog/db.changelog-master.yaml` includes ordered versioned files under `backend/src/main/resources/db/changelog/changes/` (starting from `001-initial-schema.yaml`, which uses `preConditions` with `onFail: MARK_RAN` so existing pre-Liquibase databases adopt tracking cleanly via `DATABASECHANGELOG`).
- **Four-step workflow for schema changes:**
  1. Update the JPA `@Entity` class in `backend/src/main/java/com/skillpilot/backend/domain/`.
  2. Add a new numbered YAML file `NNN-short-description.yaml` under `db/changelog/changes/` with a unique `(id, author)` changeset and explicit types/constraints.
  3. Append an `include` entry to `db.changelog-master.yaml`.
  4. Start the backend (`./gradlew bootRun` or `./gradlew test`) so Liquibase applies the migration and Hibernate `validate` confirms exact JPA-to-SQL parity.
- **Immutability rule: Never** edit an already committed and executed changeset (Liquibase checksum validation will fail startup); always append a new corrective changeset instead.

9 Real-Time Auto-Update Architecture (SSE)

When an external AI coach (via MCP or REST) persists mastery or switches the active goal, the Cockpit WebGUI updates immediately without manual page reloads through **Server-Sent Events (SSE)** (Client-State Sync).

9.1 Backend Event Bus, SSE Endpoint, and React Reconnection Hook

```
LearnerService write -> LearnerStateChangedEvent
  -> SseService (SseEmitter map + ~25s heartbeat)
  -> GET /api/ui/updates/{skillpilotId} (text/event-stream)
  -> useLearnerUpdates(skillpilotId) -> refreshMastery() / state reload
```

- **Spring event publication:** State mutations in `LearnerService.java` publish `LearnerStateChangedEvent` (`MASTERY_UPDATE`, `GOAL_UPDATE` / `ACTIVE_GOAL_UPDATE`).
- **SSE controller & emitter management:** `UpdateController` exposes `GET /api/ui/updates/{skillpilotId}` (`text/event-stream`), backed by `SseService` maintaining per-learner `SseEmitter` registrations, automatic cleanup on completion/timeout/IO error, and periodic `~25s heartbeat` comments so reverse proxies (Nginx/Cloudflare) keep idle streams open.
- **Frontend hook (`useLearnerUpdates`):** Opens a native browser `EventSource`, triggers `refreshMastery()` and active-goal state reloads on incoming events, and reconnects with bounded exponential backoff on network drops.
- **Scaling boundary:** In-memory NIO `SseEmitter` fan-out handles single-node deployments with minimal overhead; horizontal multi-instance deployments require an external pub/sub relay (such as Redis Pub/Sub or PostgreSQL `LISTEN/NOTIFY`) so events broadcast across all backend replicas.