SkillPilot Production Handbook · Volume V

Production Pipelines for Skill Graphs, Packages, and Exams

Repeatable Blueprints for Skill Graph Authoring, Secure Package Provisioning,

Subject-Export ZIPs, MEM/FWU Core Roundtrips, and Reviewed Exam Production

Scope of this handbook

This volume consolidates SkillPilot's repeatable production and export pipelines: the eight-step blueprint for authoring `SkillLandscape` JSON graphs, secure quarantine, content-addressed installation, and CAS activation/rollback for `full-standalone-v1` packages, the reproducible legacy subject-export ZIP, the Core-first MEM/FWU ontology roundtrip (`DPK-008c/008d`), and both general and Sek-I-specific exam production workflows (`exam Data`).

As of 11 November 2026 · License: Apache-2.0 (technical production and export documentation; see LICENSING.md).

Ephemeral build outputs under tmp/ and exam calibration reference pools are never committed to Git.

Contents

1 Skill Graph Production Pipeline **3**
1.1 Inputs and Steps 0–2: Extraction, Outline, and JSON Generation 3
1.2 Steps 3–7: Relations, Reference Implementations, Validation, and Registration . 3

2 Secure Curriculum Package Provisioning **5**
2.1 Six-Step Security Model and Store Layout 5
2.2 Operational Sequence, Runtime Configuration, and Self-Test 5
2.3 Deliberate Boundaries 6

3 SkillPilot Subject Export Package **7**
3.1 Build, Independent Validation, Link Audit, and CI Release Gate 7
3.2 Package Layout, Source/Memory Evidence, License Categories, and New Subjects 7

4 MEM/FWU Core Roundtrip Pipeline **9**
4.1 Two Coexisting Lanes (`DPK-007a/008c/008d` vs. Slim Pilot) 9
4.2 Core-First Mapping Rules and Image Sidecars 9
4.3 Production and Pilot Commands 10

5 Exam Example Production Pipeline **12**
5.1 Directory Layout and Blueprint Schema 12
5.2 Ten-Step Workflow (Step 0 through Step 9) 12

6 Sek-I Mathematics Exam Production Pipeline **14**
6.1 Source Hygiene and Work-Package Layout 14
6.2 Ten-Step Gated Workflow and Release Rule 14

1 Skill Graph Production Pipeline

This chapter defines the general, repeatable, auditable blueprint for turning any authoritative or institutional curriculum source into a clean `SkillLandscape` JSON graph under `curricula/<...>/json/<file>.json` (or `curricula/DE/Gymnasium/canonical/`; for exam authoring, see 5).

1.1 Inputs and Steps 0–2: Extraction, Outline, and JSON Generation

Inputs comprise (1) a normalized plain-text source extract capturing topics, outcomes, prerequisites, grading scope, and source URL/locator, (2) a human-curated outline separating root, clusters, and measurable atomic goals, plus optional (3) sibling landscapes for consistent naming and ID patterns:

- **Step 0 (Extract source):** Normalize the official curriculum into clean plain text (no HTML) with an explicit URL/page reference.
- **Step 1 (Structured outline):** Create a root node, group phases/topic fields into clusters, and cut competencies into measurable **atomic goals** (1–3 tasks, phrased as "The learner can ..."). Include **motivation anchors** and **phase-local autonomy clusters** of ordinary exam character (`Übungen E-Phase`, `Übungen Q1`–`Q4`, plus optional `Übungen Prozesskompetenzen`), keeping global Abitur/final-exam branches separate as an additional assessment layer.
- **Step 2 (Encode JSON):**
  - **Stable IDs:** Project-namespaced UUID v5 or deterministic mapping from `(landscapeId, shortKey)` (`shortKey` is optional, unique within `landscapeId`; repeating the same `(goal Id, shortKey)` pair across locale files is valid, reusing one `shortKey` for distinct `goalIds` is forbidden).
  - **Required fields:** Top-level `landscapeId`, `title`, `description`, `locale`, `subject`, `goals`; goal-level `id`, `title`, `description`, `weight`, `requires`, `contains`, `dimensionTags` (at least `phase`; prefer `area`, `topicCode`, `demandLevel`, `processCompetencies`, `guidingIdeas`; optional `tags`, `core`, `examples`, `sourceRef`, `resourceLinks`).
  - **Localization, assets & links:** Multilingual landscapes keep separate meaning-aligned locale JSONs; co-located images use relative Markdown paths. `sourceRef` holds canonical provenance, while `resourceLinks` holds ordered learning links (broad overviews on root/clusters, goal-specific practice/visualization links on atomic nodes).

1.2 Steps 3–7: Relations, Reference Implementations, Validation, and Registration

- **Step 3 (`contains` hierarchy):** Form an acyclic DAG (cluster → subgoals) with shallow, meaningful grouping.
- **Step 4 (`requires` sequencing):** Author **atomic-to-atomic** prerequisites without transitive redundancy (Graph Definition, Graph Validation Rules); inherited cluster `requires` remain a transition mechanism only. Every learner-facing atomic goal should lie on at least one teachable route from a motivation anchor to a phase-local autonomy goal. Two retained Hessen snapshots under `curricula/DE/Gymnasium/input/HE/upper-secondary/source-json/` serve as practical authoring benchmarks: `DE_HES_S_GYM_2_PHYSIK.de.json.snapshot` and `DE_HES_S_GYM_2_MATHEMATIK.de.json.snapshot` (local `Übungen E-Phase/Q1`–`Q4` plus `Übungen Prozesskompetenzen`, cleanly separated from global Abitur containers; live runtime landscapes reside under `curricula/DE/Gymnasium/canonical/`).
- **Step 5 (Validation & manual reviews):**

```
cd app
npm run validate:graph
```

Complement structural checks with manual atomicity (Atomic Review) and prerequisite (Requires Review) passes.

- **Step 6 (Register curriculum):** Add root curricula to `curricula/curriculum_manifest.json` or attach sub-curricula via parent `contains`.
- **Step 7 (QA checklist):** Clear titles/descriptions testable in 1–3 tasks, canonical atomic `requires`, acyclic `contains`/`requires`, closed motivation-to-local-autonomy routes per phase, stable IDs, consistent metadata (`phase`, `area`, `level`, `core`, `weight`), and aligned localization.

2 Secure Curriculum Package Provisioning

This runbook documents local installation, verification, CAS activation, and rollback for finished `full-standalone-v1` JSON packages. The provisioner (`scripts/provision_curriculum_package.py`) is the sole writer to the package store; the SkillPilot runtime reads only the exact pinned active lock and never mutates the store.

2.1 Six-Step Security Model and Store Layout

An incoming ZIP is never consumed directly as a runtime source:

1. Copy the unmodified input file with stable file identity into private quarantine while computing its outer SHA-256 and byte length.
2. Invoke the independent finished-package validator v2 in a separate process.
3. Stream-extract only the exact verified archive inventory into a private staging directory.
4. Re-verify manifest, `SHA256SUMS`, every manifest record, and the complete extracted file tree.
5. Atomically promote object directory, validator report, and install record under `<outerZipSha256>`.
6. Switch the active lock only through a separate, explicit compare-and-swap (CAS) step.

Symlinks, special files, path collisions, ZIP64, data descriptors, nested archives, unexpected files, and profile violations fail closed while leaving the previously active lock untouched.

```
<store>/
  .provisioner.lock
  quarantine/ & staging/
  objects/sha256/<outerZipSha256>/<archiveRoot>/
  validation-reports/<outerZipSha256>.json
  install-records/<outerZipSha256>.json
  locks/active.json & locks/history/<lockSha256>.json
```

Promoted `objects/`, validation reports, and install records are immutable; directory-scan discovery and unpinned `latest` selection do not exist.

2.2 Operational Sequence, Runtime Configuration, and Self-Test

```
python3 -B scripts/provision_curriculum_package.py --help
python3 -B scripts/provision_curriculum_package.py activate --help
python3 -B scripts/provision_curriculum_package.py rollback --help
python3 -B scripts/provision_curriculum_package.py self-test
```

Standard sequence: (1) `install`, (2) `verify`, (3) `activate` passing `--expected-active-sha256` (`none` for initial activation, otherwise the exact SHA-256 of the observed lock; entries sort canonically by `packageId`, rejecting duplicate IDs, incompatible consumer ranges, or conflicting definitions), (4) `status`, and (5) CAS `rollback` to a historical lock when needed.

Only after activation is the backend configured under a separate read-only OS identity (never falling back to repository, classpath, or network sources):

```
skillpilot:
  curriculum:
    source: package
```

```
    consumer-version: 0.1.0
    packages:
      store-directory: /srv/skillpilot/curriculum-packages
      active-lock: locks/active.json
```

2.3 Deliberate Boundaries

- Technical installation validity never grants publication, source, or redistribution approval; cryptographic publisher signatures arrive in `DPK-011`.
- `embedded-fragment` remains fail-closed until cross-subject closure and consumer lanes complete.
- Checkout-free operation (15 functional checks under `--clearenv`, loopback-only networking, and Catalog-404 fail-closed probe) has been proven since `DPK-007` through this exact store path (Curriculum Package Readiness).

3 SkillPilot Subject Export Package

This pipeline builds a reproducible legacy subject-export ZIP (`legacy-subject-export`, target status `not-ready-legacy`, `standaloneProfileReady: false` under Curriculum Package Readiness) for one subject landscape as a validated publication and roundtrip carrier (4), without claiming standalone `full-standalone-v1` runtime operation (Release Model Conformance). Supported `DE/Gymnasium` presets currently cover all `M5+` subjects: `Biologie`, `Chemie`, `Deutsch`, `Geschichte`, `Informatik`, `Latein`, `Mathematik`, `Physik`, `Politik und Wirtschaft`, and `Wirtschaftswissenschaften`.

3.1 Build, Independent Validation, Link Audit, and CI Release Gate

Single-subject export, batch export of all `M5`/`M6` subjects (`tmp/exports/m5-subject-export-summary.{json,md}`), independent ZIP validation (`tmp/exports/validation/` + `tmp/exports/readiness/`), live source-URL reachability audit (`--fail-on-broken` for strict pre-release checks), and reproducibility proof (`SOURCE_DATE_EPOCH`, uncompressed sorted entries with mode `0644`):

```
cd app
npm run export:subject-package -- --subject Mathematik --version 0.1.0
npm run export:m5-subject-packages -- --version 0.1.0
npm run export:subject-packages:validate -- --dir tmp/exports
npm run export:subject-packages:audit-links -- --dir tmp/exports
npm run export:subject-release-gate -- --version 0.1.0 --strict-source-links
npm run export:subject-release-gate:ci
npm run export:m5-subject-packages:check-reproducible -- --version 0.1.0
SOURCE_DATE_EPOCH=1767225600 npm run export:subject-package -- \
  --subject Mathematik --version 0.1.0
```

`export:subject-release-gate:ci` runs weekly/manually in `.github/workflows/owl-ci.yml` passing `--enforce-committed-quality-status --enforce-clean-source-tree` and emits release-gate and publication indexes under `tmp/exports/{release-gate,publication}/`. For reviewer handoff, `export:subject-package:trace-source` resolves source or canonical goal IDs directly from any finished ZIP:

```
cd app
ZIP=../tmp/exports/skillpilot-de-gymnasium-mathematik-v0.1.0.zip
npm run export:subject-package:trace-source -- \
  --zip "$ZIP" --source-goal-id he-math-sekii-e-1-b01-a01-376ffbbc
npm run export:subject-package:trace-source -- \
  --zip "$ZIP" --canonical-goal-id 502ecaa7-cca6-5c51-a1cc-da09a7b2382c
```

3.2 Package Layout, Source/Memory Evidence, License Categories, and New Subjects

Every archive extracts within Windows `MAX_PATH` limits under `skillpilot-de-gymnasium-<subject>-v<version>/` free of repository-local paths or learner state (the sibling `*-release-report.md` stays outside the ZIP):

- **Domain & source payloads:** `data/canonical/`, `data/views/`, `data/mappings/` (all 16 German states; partial state coverage requires explicit `--allow-missing-states`), `data/cards/` + `card-index.json`, `data/resources/goal-visualizations.json` + strictly link-selected images under `assets/goal-visualizations/`, `data/dependencies/external-goal-references.json`, and the mandatory HTTP(S) provenance bridge

`data/sources/source-index.json` ← `data/sources/source-goal-references.json` (`metadata/provenance-report.md`).

- **Quality, schemas & checksums:** `metadata/quality/memory-card-review/` (`*.md`, `*.config.json`, `*.review.jsonl`, `*.cards.review.jsonl` backing `CQR-302`), `schemas/`, `metadata/manifest.json`, `metadata/validation-report.json`, `metadata/SHA256SUMS`.
- **File-level license categories (`LICENSE.md`, `NOTICE.md`, `licenses/`):** Following `LICENSING.md`, every file in `manifest.json` carries a `licenseCategory`: `skillpilot-software-apache-2.0`, `skillpilot-data-cc-by-4.0`, `official-source-provenance-only`, `generated-package-metadata`, or `goal-visualization-ai-generated-curated` (provenance label, neither an SPDX identifier nor third-party rights clearance).
- **Reusing for new subjects without a preset:** Pass explicit `--subject`, `--subject-slug`, `--composition-dir`, and repeated `--mapping-token` arguments.

4 MEM/FWU Core Roundtrip Pipeline

This pipeline proves that a SkillPilot publication package can be transformed into a Core-first RDF/OWL bundle grounded in the FWU Lehrplan-Ontologie (`tmp/lehrplan-ontologie`, `src/ontology/components/lehrplan-core.owl`, IRI `https://w3id.org/lehrplan/ontology/lp/components/lehrplan-core.owl`) and reconstructed losslessly **without reading the original JSON or repository checkout** (Dual Release Contracts, Human Review Gates).

4.1 Two Coexisting Lanes (`DPK-007a/008c/008d` vs. Slim Pilot)

1. **Production `fwu-owl-v1` release conformance (`DPK-008a`–`008d`):** Consumes the frozen `DPK-007a` JSON artifact (`skillpilot-curriculum-de-gymnasium-mathematik-0.1.0-conformance.3.json.zip`, `1,738,161,217` bytes, SHA-256 `403cc0bc6004da549c8b9ed9fafad222fe0ddda1107806fe087cfa871a6dbcf9`, `contentDigest` `sha256:e83936aaf3645ff5f6e8132c4a801bd4bd66f55d3c0304a5deda3d6a5d194101`), builds the byte-identical 2.36-GB FWU-OWL carrier (`DPK-008c`: `2,362,455,128` bytes, SHA-256 `abab1d8aac3e9394af26c614bbf231954ba45ab11f725dd0f93f088820dc3f94`, manifest `29f308424d1aeba9095f0e800253acadcdfaca0562dfa1fc37741c77c76023b3`, `819` entries, `757` image sidecars, `824,452` triples across 8 RDF segments, 18/18 gates passed, placement key `goalId@unitId@index`), and reverse-compiles it in `DPK-008d` under isolated `bubblewrap --clearenv` (`python -I -S`, unshared network, host Rawtrace ELF audit) into two byte-identical JSON ZIPs (`1,737,571,471` bytes, SHA-256 `7dcd233dd495900f6d6bd971ff6e86bdcdcfe5701f3522ed480be8336de23195`, manifest `8d7970435431ff78743d0bf413a54bbebee009bbaafe35f1283ebcb89b4f2ff0`, 111/111 normal-form oracles; the public equivalence verdict follows in `DPK-009`).
2. **Legacy slim pilot (`tmp/exports/skillpilot-de-gymnasium-mathematik-v0.1.0.zip`):** Fast regression lane under `tmp/roundtrip/mem-fwu/.../slim/` (`declarations.nt` with exactly `526` declaration triples, `landscape.nt`, `assets.nt`, `sources.nt`, `mappings.nt`, `views.nt`, `cards.nt`, derived `bundle.nt`, `catalog-v001.xml`, `skillpilot-mem-fwu-profile.ttl`, `manifest.json`). Because the legacy comparator drops `null` fields and compares several arrays as sets, its reports remain regression evidence (`not-ready-legacy`) and never serve as the public equivalence verdict.

4.2 Core-First Mapping Rules and Image Sidecars

SkillPilot concept	Primary FWU Core representation	Application extension / reconstruction anchor
Curricular atomic goal (`curricularAtomic`)	`LP_0000336` (*Fachbezogene Kompetenzspezifikation*; parentless atoms attach to one deterministic fallback `LP_0000349` area)	`sp:LearningGoal` + `sp:AtomicGoal` (fallback area ignored during reconstruction)
Curricular cluster (`curricularArea`)	`LP_0000349` (*CE-Bereich*)	`sp:LearningGoal` + `sp:ClusterGoal`
Process axes `K1`–`K6` / guiding ideas `L1`–`L5`	27 `LP_0030265` and 5 `LP_0000268` resources; linked via `LP_0030065` (`LP_0030071`/`LP_0030072`)	`sp:referenceRole` separates authored `competencyRefs` from derived `dimensionTags`; `sp:dimensionTagsJson` ensures lossless recovery

SkillPilot concept	Primary FWU Core representation	Application extension / reconstruction anchor
Program year / stage	`LP_0000026` (`LP_2000005`–`LP_2000010`), `LP_0000047` (Sek I/II)	`sp:ProgramUnit` & placement metadata (`goalId@unitId@index`)
Curricular `contains`	`BFO_0000051` (*hat Teil*, transitive)	Direct `sp:containsGoal` prevents BFO transitivity from creating false direct children
Curricular vs. runtime `requires`	Reified `LP_0000554` (*Didaktische Voraussetzung*, from `FWU-DE/lehrplan-ontologie#9`)	`sp:didacticRequires` for practice, assessment, memory, and orientation nodes
Title / description / short key	`LP_0030056` / `LP_0030051` / `LP_0030057` → `LP_0000344`	`rdfs:label`, `dcterms:description`, `sp:shortKey`
Curricular vs. runtime goal image	`LP_0030065` to `schema:ImageObject` ordered by `LP_0000460`	`sp:GoalVisualizationReference` for runtime goals; hashed binary sidecars (never Base64 in `.nt`; caps: `64 MiB`/image, `8 GiB` total)
Sources & mappings	`LP_0000438` (*Lehrplan*), `LP_0000261` (*Curriculares Element*), `LP_0030065`	`sp:OfficialSourceDocument` for supplemental docs; `exact/partial` predicate; quality evidence stays strictly in `sp:*`

Compact JSON metadata uses `xsd:string` (avoiding custom datatypes that break OWL 2 DL; `sp:fieldState` preserves empty collections). Whereas legacy bundles quarantined OCR control characters in `sp:quarantinedRecordJson`, the `DPK-004` release compiler rejects C0 controls and unpaired UTF-16 surrogates outright. The pinned Core file (OWL Functional Syntax, `text/owl-functional`) plus application ontology, SHACL shapes (free of SPARQL/JS), and XML catalog under `contracts/curriculum-package/v1/ontology/` act as external trust roots.

4.3 Production and Pilot Commands

Pinned offline toolchain (`jsonschema 4.26.0`, `pySHACL 0.30.1`, `RDFLib 7.6.0`, `owlrl 7.6.2`, `ROBOT 1.9.10`, `HermiT 1.4.5.456`, Amazon Corretto), self-tests, and heavyweight 2.36-GB production wrappers from the repository root:

```
python3 -m pip install -r scripts/curriculum_fwu_owl_validation_requirements.txt
bash scripts/provision_pinned_robot.sh
python3 -B scripts/check_curriculum_fwu_owl_validation_tools.py \
  --report tmp/curriculum-release-model/fwu-owl-validation/tools-report.json
python3 -B scripts/validate_fwu_owl_curriculum_package.py --self-test
python3 -B scripts/reconstruct_json_curriculum_package_from_fwu_owl.py --self-test
python3 -B scripts/run_fwu_owl_reverse_compiler_hermetic.py --self-test
python3 -B scripts/validate_curriculum_fwu_owl_reverse_compilation_contract.py \
  --self-test
bash scripts/run_curriculum_fwu_owl_package_conformance.sh
bash scripts/run_curriculum_fwu_owl_reverse_conformance.sh
```

Running the fast legacy slim pilot from `app/` (semantic roundtrip, OWL 2 DL/HermiT reasoning, and optional technical line-carrier roundtrip):

```
cd app
npm run export:subject-package -- --subject Mathematik --version 0.1.0
npm run export:subject-packages:validate -- \
  --zip ../tmp/exports/skillpilot-de-gymnasium-mathematik-v0.1.0.zip
npm run roundtrip:mem-fwu:semantic
ROBOT_JAR=/abs/path/to/robot.jar npm run roundtrip:mem-fwu:owl:reason
npm run roundtrip:mem-fwu
```

5 Exam Example Production Pipeline

This chapter defines the general, repeatable blueprint for authoring quality-assured draft exam packages from official exam specifications before promoting stabilized packages into the skill graph as `examData` (1).

5.1 Directory Layout and Blueprint Schema

Subject-level control files reside under `curricula/<...>/abi/<Subject>/` (`exam-example-authoring-guide.md`, `input/references.md`, `*_exam_blueprint*.json`), with one subfolder per example package such as `Klausurbeispiel2026_1/`:

```
curricula/<...>/abi/<Subject>/Klausurbeispiel2026_1/
  checkliste.md
  <Subject>_<Region>_<Year>_Klausurbeispiel_<n>.md
  <Subject>_<Region>_<Year>_Klausurbeispiel_<n>_Musterloesung.md
  Schritt5_Abschluss_QS.md
  findings.md
  Schritt7_Finding_Bearbeitung.md
  Schritt8_ReQS.md
```

Temporary similarity or originality reports stay strictly under `tmp/`. The planning blueprint fixes coverage, selection modes, and complementarity to prior examples upfront:

```
{
  "examSpecId": "subject-region-year-example",
  "locale": "de-DE",
  "subject": "Physik",
  "frameworkRefs": ["Official exam hint", "Curriculum"],
  "selectionModes": {},
  "contentFocus": {},
  "proposalMatrix": [],
  "globalCoverageChecks": []
}
```

5.2 Ten-Step Workflow (Step 0 through Step 9)

1. **Step 0 (Register sources):** Record decrees, curriculum standards, style corpora, and prior SkillPilot examples in `input/references.md`, cleanly separating binding regulations from style references.
2. **Step 1 (Create QA checklist):** Phrase verifiable checks covering metadata, selection rules, topic coverage, cognitive level, originality, permitted aids, and task/solution/scoring consistency (`checkliste.md`).
3. **Step 2 (Create blueprint):** Plan `GK`/`LK` course variants, task-part rules, proposal matrix, and explicit complementarity/originality relative to earlier examples.
4. **Step 3 (Draft exam tasks):** Write independent, non-copied tasks with realistic contexts, clean operator wording, consistent units/data, and strictly learner-facing text.
5. **Step 4 (Draft solution & scoring):** Provide complete per-subtask derivations, transparent `BE` point mapping, explanatory steps, and explicit rounding/tolerance rules.
6. **Step 5 (Run final QA):** Record item-by-item checklist verdicts, residual risks, and release status (`draft-ready`, `release candidate`) in `Schritt5_Abschluss_QS.md`.
7. **Step 6 (Capture findings):** Turn review feedback into structured `findings.md` items (ID, source, description, priority, affected files, acceptance criterion, status).

8. **Step 7 (Process findings):** Log addressed findings, exact edits, rationale, originality/similarity checks, and any follow-up risk in `Schritt7_Finding_Bearbeitung.md`.
9. **Step 8 (Re-QA after findings):** Document finding-by-finding status deltas (`behoben`, `teilweise behoben`, `offen`), affected checklist categories, and updated release status in `Schritt8_ReQS.md`.
10. **Step 9 (Integrate into skill graph):** Only after stabilization, promote exam nodes with `examData.taskContent`, `examData.solutionContent`, and `examData.scoring` into the landscape JSON (`docs/concept/skill-graph/node-types.md`). Subject-specific `exam-example-authoring-guide.md` files instantiate this workflow with concrete sources, paths, and current step without duplicating the general pipeline.

6 Sek-I Mathematics Exam Production Pipeline

This chapter specifies the reviewed production workflow for canonical German Gymnasium Mathematics exams in Sekundarstufe I (grades 5 through 10). Each visible year scope exposes a dedicated `Prüfungen Jahrgangsstufe <n>` folder containing atomic exam-task nodes; neither a standalone `Sek-I-Abschlussaufgaben Mathematik` capstone nor a single aggregate `Übungen Sekundarstufe I` branch may serve as the learner-facing route endpoint.

6.1 Source Hygiene and Work-Package Layout

External reference tasks may be used solely for style, quality, and difficulty calibration — never copied, translated, closely paraphrased, or structurally cloned:

- Downloaded reference material stays strictly under `tmp/seki-math-exam-reference-pool/` (with a local `sources.json` or `README.md` recording URL, provider, retrieval date, year relevance, and intended use) and is **never** committed to Git.
- Preferred calibration pools: IQB Sek I / VERA-8, Landesbildungsserver / ZSL / Lehrerfortbildungsserver, institutionally curated task databases such as SMART, and official syllabus pages (avoid private worksheets with unclear rights, answer-key scrapers, or uncurated AI task lists).
- Whenever an external item inspires a task idea, record only the abstract design principle — never wording, numbers, story, diagram, or solution path.

Shared release checklist: `curricula/DE/Gymnasium/assessments/mathematik/seki/quality-checklist.md`; per-year work package layout (flat legacy files such as `j5_terminal_autonomy_v1.md` are superseded once a reviewed package is promoted):

```
curricula/DE/Gymnasium/assessments/mathematik/seki/j5/
  blueprint.md & draft_v1.md & solution_v1.md
  internal_qa_v1.md & external_review.md
  findings.md & finding_resolution.md
  re_qa.md & release_candidate.md
```

6.2 Ten-Step Gated Workflow and Release Rule

Step	Required artifact & gate criterion
0. Reference pool	At least two independent high-quality source families checked under `tmp/seki-math-exam-reference-pool/`; zero external task text committed.
1. Checklist	`quality-checklist.md` confirmed by the Product Owner or an external reviewer before drafting begins.
2. Blueprint	Year level, timing, covered canonical goal IDs, guiding ideas/process competencies, point/demand matrix, explicit non-coverage notes, and permitted aids recorded.
3. Raw draft	Age-appropriate context, accessible entry plus reasoning/checking/model-limit prompt per task, independent tasks free of hidden assumptions (`draft_v1.md`).
4. Solution & BE	Complete derivation, transparent `BE` points tied to observable student work, accepted alternative paths, and common-error notes (`solution_v1.md`).
5. Internal QA	`internal_qa_v1.md` grading every item `pass`, `fail`, `blocked`, or `not_applicable`; zero `fail`/`blocked` in release-critical sections.
6. External review	`external_review.md` with positive sign-off or concrete remediation work for every blocking finding.

Step	Required artifact & gate criterion
7. Iterate findings	`findings.md` & `finding_resolution.md`; every finding marked `resolved`, `accepted_risk`, or `out_of_scope` with rationale.
8. Re-QA & candidate	`re_qa.md` & `release_candidate.md`: checklist complete, external blockers resolved, originality and source hygiene verified.
9. Graph promotion	Update canonical exam nodes (`examData.reviewStatus = "released"`, narrow `examData.coveredGoalIds`, actually assessed `requires`), refresh source/rationale reports, and pass full CI with tasks visible under `Prüfungen Jahrgangsstufe <n>`.