SkillPilot Qualitätshandbuch · Band IVb

# CI-Architektur, Release-Verträge und Review-Lanes

Gestufte CI-Pipelines, Qualitätsdashboard, Mapping-Workbench, Standalone-Paket-

und Dual-Release-Verträge, Review-Prozesse (HR-001–HR-006, MEM/SPARQL) und Runbooks

**Geltungsbereich dieses Bandes**

Ergänzend zu Band IV (Reifegrade `M0`–`M7`, `CQR-*`, `GVR`/`APV`/`CPV`, semantische Atomizität und Champion-Modell) dokumentiert Band IVb die operative Prüf- und Release-Infrastruktur: die zweistufige GitHub-Actions- und `./run_ci.sh`-Architektur, das Qualitätsdashboard und die Mapping-Workbench, die Fail-Closed-Paketbereitschaft (`full-standalone-v1`), Release-Modell-Konformität und Dual-Release-Verträge sowie die manuellen und teilautomatisierten Review-Lanes (`HR-001`–`HR-006`, Atomic/Requires-Review, MEM-SPARQL-Audit, Quellenbegründungen und Abi-2026-DeepLink-QA).

Stand: 11. November 2026 · Lizenz: Apache-2.0 (technische CI-, Paket- und Review-Dokumentation; siehe LICENSING.md).

Maschinelle Konformität ersetzt keine menschlichen Freigabegates (HR-001 bis HR-006) und keine praktische Erprobung.

## Inhaltsverzeichnis

**1 Continuous Integration (CI) 4**
1.1 Auslöser und zweistufige Ausführung . . . 4
1.2 Jobs im Quick Tier und Complete Check . . . 4
1.3 Curriculum-, View- und Archivgrenzen sowie optionale Paket-CI . . . 5
1.4 Ubuntu-Paketquellen und Mirror-Failover . . . 6
1.5 Lokale Ausführung und Status-Snapshots . . . 6

**2 Curriculum-Qualitätsdashboard 8**
2.1 Generatoren, Artefakte und lokale Befehle . . . 8
2.2 Reifegrade `M0`–`M7`, Quellenbelege und Regelfamilie `curriculum-quality-v5` . . . 8
2.3 Mapping-Pipeline, Bundesland-Evidenz und Routenprofile . . . 9

**3 Curriculum-Mapping-Workbench 11**
3.1 Siebenteiliger Datenfluss im Repository . . . 11
3.2 Fail-Closed-Pipeline-Gates (`MAPPING-1` bis `MAPPING-3`) . . . 11
3.3 Zweistufiges Interaktionsmodell und Prüfinterpretation . . . 12

**4 Curriculum-Paketbereitschaft (`full-standalone-v1`) 13**
4.1 Statusvokabular und Exit-Codes . . . 13
4.2 Konformitätssuiten, Bau und hermetischer Paket-Consumer . . . 13
4.3 Offline-Vertrauensmodell, Review-Trennung und Umsetzungsstand (`DPK-004a`–`DPK-008d`) . . . 14

**5 Curriculum-Release-Modell-Konformität 16**
5.1 Überblick und vertrauenswürdige Eingaben . . . 16
5.2 Erzeugte Modellstruktur, Digests und Publikationsevidenz . . . 16
5.3 Prüfkommandos und gehosteter PDF-Modus . . . 17

**6 Duale Curriculum-Release-Verträge 19**
6.1 Verbindliche Vertragsinvarianten . . . 19
6.2 Ausführbares Gate und Abgrenzung . . . 19

**7 Prüfung der Requires-Relationen 21**

**8 Atomizitäts-Reviewprozess 22**
8.1 Geltungsbereich und Arbeitsdefinition . . . 22
8.2 Siebenstufiger Review-Ablauf und Entscheidungsdatei . . . 22

**9 Voraussetzungs-Reviewprozess (`requires`) 24**
9.1 Übergangssemantik vs. kanonischer Zielzustand . . . 24
9.2 Sechsstufiger Prüf- und Reparaturablauf . . . 24

**10 Menschliche Review-Gates für Curriculum-Releases (`HR-001`–`HR-006`) 26**
10.1 Überblick der sechs unabhängigen Gates . . . 26
10.2 Fach-, Rechte- und Quelltext-Gates (`HR-001` bis `HR-004`) . . . 26
10.3 Ontologie-Sign-off und Stable-Promotion (`HR-005` & `HR-006`) . . . 27

**11 MEM-SPARQL-Konsistenzaudit 28**
11.1 Maßgebliche Quellen, generierte Artefakte und Ausführung . . . 28
11.2 Fingerprint-gebundenes Review-Ledger und Erweiterungskriterien . . . 28

**12 MEM-SPARQL-Konsistenz-Runbook** **30**
12.1 Audit ausführen und Queue inspizieren . . . . . 30
12.2 Entscheidungen eintragen, Diagnostik deuten und lokal verifizieren . . . . . 30

**13 Runbook für Lernziel-Quellenbegründungen** **32**
13.1 PoC-, Zielmengen- und MEM/FWU-Showcase-Kommandos . . . . . 32
13.2 Öffentlicher Laufzeitindex (Repository- vs. Paketmodus) & Deployment-Smoke . 32
13.3 Review-Checkliste und Folge-Issues . . . . . 33

**14 Abi-2026 Mathe DeepLink MVP (BlueSky)** **34**
14.1 Linkstruktur und Cockpit-Vorkonfiguration . . . . . 34
14.2 Kampagnen-Events und QA-Prüfung (3 Test-Links) . . . . . 34

# 1 Continuous Integration (CI)

Dieses Kapitel dokumentiert den zweistufigen CI-Workflow in `.github/workflows/ci.yml`, den optionalen Paket- und Ontologie-Workflow in `.github/workflows/owl-ci.yml` sowie die lokale Suite `./run_ci.sh`.

## 1.1 Auslöser und zweistufige Ausführung

Der Workflow `.github/workflows/ci.yml` startet bei:

- Push auf `main`,
- Pull Requests gegen `main` (beim Öffnen, Aktualisieren, Wiederöffnen oder Wechsel auf `ready for review`),
- manuellem Start (`workflow_dispatch`).

Reine Dokumentations- und temporäre Änderungen (`README.md`, `AGENTS.md`, `docs/**`, `sandbox/**`, `tmp/**`) lösen `ci.yml` nicht aus; für die Dokumentation greift der eigene Workflow `.github/workflows/docs_checks.yml`.

Die Ausführung ist in zwei Stufen gegliedert, deren Struktur durch `scripts/ci_tiers.test.mjs` in `quick-checks` erzwungen wird:

- **Quick Tier:** Läuft bei jedem Trigger einschließlich Draft-Pull-Requests. Die Jobs warten auf keinen anderen Job und liefern daher sofortiges Frühfeedback.
- **Complete Check:** Läuft bei Pushes auf `main`, manuellen Starts und nicht als Draft markierten Pull Requests parallel über alle vollständigen Prüfjobs.

Verbindliche Stufenregeln:

- Der Complete Check enthält jede erforderliche Prüfung eigenständig; der Quick Tier wiederholt schnelle Prüfungen nur vorab und ersetzt keinen Vollcheck.
- Ein Draft-PR erhält nur den Quick Tier (Complete-Check-Jobs erscheinen als `skipped`). Beim Markieren als `ready for review` startet der Complete Check für denselben Stand; da Git Hub keine Drafts mergt, besitzt jeder mergebare PR einen vollständigen Lauf auf seinem aktuellen Head.
- Weder Quick Tier noch Complete Check filtern einzelne Jobs anhand geänderter Dateipfade.
- Abgespaltene Teiljobs behalten Runner-Image, Werkzeuge und wiederhergestellte Zielbuch-Artefakte ihres Ursprungsjobs bei.

## 1.2 Jobs im Quick Tier und Complete Check

| Quick-Tier-Job | Geprüfter Umfang |
|---|---|
| `action-regression-ci` | Spring-gehosteter Custom-GPT-Action-Regressionsclient und Deployment-Entrypoint-Guards |
| `openai-mcp-app-ci` | OpenAI-MCP-App-Verträge, Plugin-Release- und Submission-Prüfungen |
| `claude-connector-v1-ci` | Claude-MCP-Apps, Plugin-Paket und Direct-Install-Publikation |
| `openai-v1-public-edge-ci` | OpenAI-V1-Public-Edge- und mTLS-Verifikationsskripte |
| `quick-checks` | CI-Strukturtests sowie frühe Läufe von ESLint, TypeScript-Typprüfung, `validate:graph` und `validate_schemas.py` |
| `backend-quick-checks` | CI-Runner-Tests und Kompilierung der Backend-Haupt- und Testquellen (ohne Testausführung) |

Im Complete Check baut und verifiziert `goal-book-publication` einmalig alle registrierten Lernzielbücher; alle mit `ja` markierten Folgejobs stellen dieses Artefakt desselben Laufs wieder her:

| Complete-Check-Job | Zielbuch | Geprüfter Umfang |
|---|---|---|
| `goal-book-publication` | – | Bau und Verifikation aller registrierten Lernzielbuch-Publikationen |
| `frontend-ci` | ja | Anwendungslogik auf Fixtures, Paket-/Laufzeitadapter, Lernzielauswahl, GPT-Instruktionslimits, deterministisches ZIP-Primitiv, ESLint, `npm run build:application` (nur App gegen eingecheckte Assets, im Unterschied zum integrierten `npm run build`) und Prüfungen auf dem Build |
| `goal-book-contracts` | ja | `test:goal-book-pipeline` |
| `graph-validation` | ja | Skill-Graph-, View-, Quellen-, Review- und Status-Gates |
| `curriculum-assessment-proofs` | ja | Beweise transitiver Minimalität sowie Assessment-, Prüfungs- und Routentests |
| `curriculum-integrated-build` | ja | Laufzeitassets, Visualisierungsprüfungen, integrierter App-Build und reines Paket-Frontend |
| `curriculum-package-contracts` | ja | Paketverträge und Selbsttests, Schemata, Ziel-IDs, Archiv- und Legacy-Referenzvalidatoren |
| `backend-ci` | ja | Alle Backend-Testklassen außer den Voraussetzungs-Reduktionsbeweisen in drei disjunkten parallelen Test-JVMs (`scripts/run_backend_ci.py`; fehlgeschlagene Vorläufertests werden zuerst wiederholt) |
| `backend-reduction-proofs` | nein | Voraussetzungs-Reduktionsbeweise in zwei parallelen Partitionen (alle Fächer ohne Physik sowie die Physik-Methode isoliert); `./gradlew check` ohne Optionen führt weiterhin die Gesamtsuite seriell aus |

### 1.3 Curriculum-, View- und Archivgrenzen sowie optionale Paket-CI

Die vier Curriculum-Jobs validieren gemeinsam Code, Verträge, Modelle, Reviews, Schemata und eingecheckte Statusdateien, materialisieren im regulären Commit-Lauf jedoch **kein** reales 1,7-GB-Curriculum-ZIP:

- **Graph- und Projektionsregeln:** `validate:graph` läuft standardmäßig strikt (`GVR-*` blockierend; Warnmodus lokal über `VALIDATE_GRAPH_STRICT_RULES=0 npm run validate:graph`). `validate:view-filters` prüft in der CI die 17 kanonischen DE-Gymnasium-Landschaften (`Mathematik`, `Physik`, `Chemie`, `Biologie`, `Informatik`, `Deutsch`, `Englisch`, `Französisch`, `Griechisch`, `Chinesisch`, `Geschichte`, `Politik und Wirtschaft`, `Musik`, `Latein`, `Spanisch`, `Wirtschaft`, `Overview`; lokal erweiterbar per `APPLICABILITY_VALIDATION_SCOPE=all`). Akzeptierte `APV-201`-Restwarnungen stehen in `docs/qa-ci/applicability-accepted-warnings.json`; `APV-202` ist rein diagnostisch.
- **Dauer-, Quellen-, Dokumentations- und Merkkarten-Gates:** Geprüft werden die hessische Mathematik-Sek-I-Dauerprojektion G8/G9 (`check:he-math-duration-projection`), die sichtbare Quellenabdeckung für Mathematik und Physik (`quality:source-coverage-audit:check`), explizite Kompositionsansichten (`validate:composition-views`), Dokumentationshinweise, Statusregister, Links und Indizes (`check:generated-doc-notices`, `check:generated-status-registry`, `check:docs-links`, `check:docs-indexes`) sowie alle konfigurierten Merkkarten-Ledger (`quality:memory-card-review:check:all`, wobei fehlende `CQR-302`-Konfiguration `M6`, nicht aber `M5` blockiert).

- **Archiv- und Legacy-Referenzgrenzen:** Die Validatoren sichern die hessischen Bestandsgrenzen unter `curricula/DE/Gymnasium/input/HE/abi` (`curricula/DE/Gymnasium/input/HE/retained-asset-registry.json` und `curricula/DE/Gymnasium/provenance/hessen-upper-secondary-retirement-registry.json`), die Sek-I-Grenzen unter `curricula/DE/Gymnasium/input/HE/lower-secondary` (`curricula/DE/Gymnasium/provenance/hessen-lower-secondary-retirement-registry.json`) sowie die Artefakte der hessischen Chemie-Abiturpipeline 2026 (`slot_matrix.json`, `coverage_requirements.json`, `task_bank.json`).
- **Optionale Paket- und FWU-OWL-CI (`.github/workflows/owl-ci.yml`):** Läuft wöchentlich oder manuell auf frischen Runnern für das reale 1,7-GB-Standalone-JSON-Paket samt hermetischem Consumer, alle Fachexport-ZIPs und die Core-first-FWU-OWL-/Roundtrip-Lane. Die Produktivlaufzeit nutzt ausschließlich JSON-Pakete und benötigt kein OWL; der verpflichtende JSON-Release-Modell-Check verifiziert lediglich den im Build-Profil gepinnten FWU-Core-Commit und Datei-Hash ohne RDF-/ROBOT-/HermiT-Lauf. Die realen 2,36-GB-FWU-OWL-Wrapper bleiben explizite Release-Operationen.

### 1.4 Ubuntu-Paketquellen und Mirror-Failover

Jeder Job mit Betriebssystem-Paketinstallation führt vorab `.github/actions/ubuntu-apt` aus und beschränkt APT strikt auf offizielle Ubuntu-Quellen (ohne Chrome- und Microsoft-Repositories des Runner-Images):

- **Mirror-Listen-Failover:** Jede Ubuntu-Stanza liest `/etc/apt/skillpilot-ubuntu-mirrors/` (`archive.txt` bevorzugt `https://archive.ubuntu.com/ubuntu/`, `security.txt` bevorzugt `https://security.ubuntu.com/ubuntu/`, jeweils mit dem Azure-Mirror `http://azure.archive.ubuntu.com/ubuntu/` an zweiter Stelle). Der APT-Mirror-Transport schwenkt fehlgeschlagene oder länger als 30 s blockierte Dateien auf die Alternativquelle um; Wiederholungen auf derselben Quelle sind deaktiviert (`Acquire::Retries "0"`).
- **Vorab-Probe auf `InRelease` & Signaturprüfung:** Vor jedem `apt-get update` (auch bei verschachtelten Playwright-Aufrufen) prüft `/usr/local/sbin/skillpilot-order-ubuntu-mirrors` `InRelease` jeder Quelle (maximal 20 s), vergibt Priorität `1` an die erste antwortende Quelle und protokolliert `SkillPilot Ubuntu mirror order: …`. Suites, Komponenten und `Signed-By` bleiben unverändert, sodass jeder Mirror gegen den Ubuntu-Keyring und signierte Index-Hashes verifiziert wird; ci_ubuntu_apt.test.mjs sichert diese Richtlinie offline gegen ein lokal signiertes Fixture-Repository ab.

### 1.5 Lokale Ausführung und Status-Snapshots

Vom Repository-Root steuert `./run_ci.sh` die lokalen Prüfbereiche (unbekannte oder mehrere Argumente enden mit Exit-Code `2`):

```
./run_ci.sh
./run_ci.sh all
./run_ci.sh application
./run_ci.sh curriculum
./run_ci.sh package
./run_ci.sh owl
./run_ci.sh full
```

- `application`: Custom-GPT-Action-Regression, Fixture-basierte Frontend-Logik, GPT-Instruktionslimits, Lint, `build:application` und isolierte Gradle-Backend-Checks.

- `curriculum`: Graph-/Daten-/Review-/Asset-Gates, Schemata und deterministische JSON-Release-Modell-Konformität (ohne reales 1,7-GB-ZIP).
- `package`: Optionales reales Standalone-JSON-ZIP, unabhängige Paketvalidierung, sichere Provisionierung/Aktivierung, Readiness-Evaluation und hashgebundener hermetischer Consumer.
- `owl`: Optionale FWU-OWL-Builder/Verträge, Reverse-Compiler-Tests, semantischer MEM/FWU-Roundtrip und OWL-2-DL-/HermiT-Validierung.
- `all` (Standard ohne Argument): Alle verpflichtenden Application- und Curriculum-Prüfungen genau einmal (immer abschließend bei bereichsübergreifenden Laufzeitänderungen ausführen).
- `full`: `all` zuzüglich der optionalen Lanes `package` und `owl`.

Optionale lokale Status- und MEM-Snapshots:

```
cd app
npm run quality:curriculum-status
npm run quality:mem-sparql-consistency
```

Der erste Befehl aktualisiert curriculum-quality-status.json und `.md` für `/quality-dashboard`; der zweite schreibt die nicht-blockierenden MEM/FWU-Audit- und Review-Queue-Artefakte unter `docs/qa-ci/status/` (siehe 12).

## 2 Curriculum-Qualitätsdashboard

Dieses Kapitel spezifiziert die persistierte Qualitätsstatus-Schicht für die lokale schreibgeschützte Workbench-Ansicht unter `/quality-dashboard` (detaillierte Fachsemantik der Reifegrade und `CQR-*`-Regeln siehe Qualitätshandbuch Band IV).

### 2.1 Generatoren, Artefakte und lokale Befehle

Das Dashboard implementiert keine eigene Validierungslogik in React, sondern liest ausschließlich generierte Status-Snapshots über `/__quality-dashboard/status` und `/__quality-dashboard/file?path=docs/qa-ci/status/<file>.md`:

- **Hauptstatus:** `app/scripts/generateCurriculumQualityStatus.ts` schreibt curriculum-quality-status.json und curriculum-quality-status.md.
- **Merkkarten-Review (`CQR-302`):** `app/scripts/memoryCardReview.ts`, `runMemoryCardReviewConfigs.ts` und `reportMemoryCardReviewRollout.ts` prüfen `curricula/DE/Gymnasium/quality/memory-card-review/*.review.jsonl` sowie `*.cards.review.jsonl` und schreiben die Berichte `docs/qa-ci/status/memory-card-review-*.md` (aktuell in CI abgeschlossen und erzwungen für Mathematik, Physik, Chemie, Biologie, Deutsch, Geschichte, Informatik, Latein, Politik und Wirtschaft sowie Wirtschaftswissenschaften).
- **Sanierungs-, Quellen- und MEM-Berichte:** `reportM0RemediationPlan.ts` (`m0-remediation-plan.md`), `reportEnglishRemediationPilot.ts` (`english-remediation-pilot.md`), `generateCurriculumSourceCoverageAudit.ts` (`curriculum-source-coverage-audit.{json,md}`) und der bewusst nicht-blockierende `generateMemSparqlConsistencyAudit.ts` (`mem-sparql-consistency-audit.{json,md}`).

Aktualisierung der lokalen Snapshots und Bootstrap einer sichtbaren offenen Merkkarten-Queue (`needs_developer_review` ohne inhaltliche Freigabe; `pilot-dossier` dient der unverbindlichen Vorab-Triage):

```
cd app
npm run quality:curriculum-status
npm run quality:memory-card-review:report:all
npm run quality:memory-card-review:check:all
npm run quality:memory-card-review:rollout
npm run quality:memory-card-review:pilot-dossier
npm run quality:m0-remediation
npm run quality:english-remediation-pilot
npm run quality:source-coverage-audit
npm run quality:mem-sparql-consistency
npm run quality:memory-card-review -- --mode=bootstrap \
  --config=curricula/DE/Gymnasium/quality/memory-card-review/<id>.config.json
```

### 2.2 Reifegrade `M0`–`M7`, Quellenbelege und Regelfamilie `curriculum-quality-v5`

Das Dashboard weist jedem Curriculum konservativ einen Reifegrad `M0`–`M7` zu und zeigt zusätzlich den Gate-Streifen `M5-M7 review gates` (`CQR-301`, `CQR-401`, `CQR-501` für Kernstufe `M5`, `CQR-302` für `M6`, `CQR-303` für `M7`). Menschliche Champion-Erprobungen bleiben eine orthogonale Aussage (Qualität & Erprobung). Amtliche Quellen werden zweischichtig geführt: dauerhaft eingecheckt sind strukturierte `sourceDocument`/`sourceDocuments`-Referenzen mit offiziellen HTTP(S)-URLs und lesbaren extrahierten Inventaren; lokale Arbeitskopien (`curricula/**/*.pdf`) sind regulär

per Git ignoriert, sodass ein grünes Quellen-Badge auch bei `0/n` lokalen PDFs gültig ist, sofern alle `n/n` offiziellen URLs und Inventare vorliegen.

| Regel | Ziel | Bedeutung in `curriculum-quality-v5` |
|---|---|---|
| `CQR-000` | `M1` | Quelleninventare lesbar, mit offiziellen HTTP(S)-URLs verknüpft und extrahierte Ziele im Membership-/Closure-Ledger registriert. |
| `CQR-001` | `M0` | Basis-Graphintegrität: IDs, lokale/globale Referenzen, Selbstreferenz-Schutz und direkte DAG-Prüfung. |
| `CQR-002` | `M0` | Explizite `type`-Metadaten stimmen mit struktureller Atom-/Cluster-Klassifikation überein. |
| `CQR-003` | `M2` | Bundesland-Abdeckung: volle Quellenbindung aller quellenrelevanten View-Atome, keine unbelegten Zuordnungen, keine fehlenden registrierten Originalziele. |
| `CQR-004` | `M2` | Kursniveau-Konsistenz: Quellziele (`GK_LK`, `LK`, `unspecified` mit Standard `GK_LK`) mappen nur auf kompatible `GK`/`LK`-Ziele. |
| `CQR-101` | `M3` | Effektive Routenabdeckung über $R_{\text{effective}}$ vom Motivationsanker über Atomziele bis zur terminalen Autonomie. |
| `CQR-102` | `M3` | Direkte atomare Routenabdeckung über explizite atomare `requires`-Kanten. |
| `CQR-103` | `M3` | Keine verbleibenden Cluster-`requires` für die reguläre Sequenzierung im Routenscope. |
| `CQR-201` | `M4` | Terminale Autonomieziele besitzen `examData` (oder eine explizit geprüfte Ausnahmekonvention). |
| `CQR-301` | `M5` | Ledger der semantischen Atomizität für fachliche Blattziele sind vollständig, aktuell und aufgelöst. |
| `CQR-302` | `M6` | Merkkarten-Ledger entscheiden explizit über Memorierungsbedarf, binden jede aktive Primärkarte und sichern Sichtbarkeit in Lernansichten. |
| `CQR-303` | `M7` | Zentraler Bericht aus `reportDeepUnderstandingRollout.ts` beweist die exakte Schnittmenge der Gates D/P/A/M/V auf allen aktuellen `curricular Atomic`-Zielen (technische Bild-Deferrals zählen nie als erfüllt; `human Approved`/`humanIssueIdentified` erfüllen oder blockieren Gate V/`M7` nicht). |
| `CQR-401` | `M5` | Mindestens eine lernendenseitige Kompositionsansicht existiert. |
| `CQR-501` | `M5` | Applicability-Warnungen sind sauber in aktive, aktuell akzeptierte und veraltet akzeptierte Zähler getrennt (`active = 0`). |

### 2.3 Mapping-Pipeline, Bundesland-Evidenz und Routenprofile

Unabhängig von `M0`–`M7` zeigt das Dashboard den Fortschritt der `source-extraction`-Pipeline je Quelllandschaft (`complete`, `incomplete`, `blocked`; fehlende Extraktionen erscheinen als `0/3` mit offenem `MAPPING-1`):

1. **`MAPPING-1:`** Amtliche Lehrplanpassagen vollständig extrahiert.
2. **`MAPPING-2:`** Quellziele aus diesen Passagen gebildet.
3. **`MAPPING-3:`** Quellziele auf kanonische SkillPilot-Ziele abgebildet (für Hessen Mathematik Sek II sind `E.1`–`E.7`, `Q1.1`–`Q1.5`, `Q2.1`–`Q2.5`, `Q3.1`–`Q3.5`, `Q4.1`–`Q4.3` mit `316/316` Quellzielen abgeschlossen; für Hessen Mathematik Sek I alle `426/426` Quellziele aus KC 7.3, KC 7.1/7.2/6 und G9-Themenlinien).

Für **`CQR-003` (Bundesland-Abdeckung)** zählen ausschließlich quellenrelevante atomare Ziele lernendenseitiger Kompositionsansichten (ohne Memory/SRS-, Übungs-, Assessment-, Motivations-, Orientierungs- und `examData`-Knoten). Als direkter Lehrplannachweis gelten **nur** `provenance`, geprüfte `mapping`-Kanten (einschließlich `partial`, die als Passgenauigkeitswarnung

sichtbar bleiben) und explizit akzeptierte Surrogat-Einträge für reale Voraussetzungslücken — **niemals** manuelle `override`-Overlays, `child-union` oder automatische `requires-closure`.

Aktuell erforderliche Routenprofile und lokaler Prüfablauf (z. B. für den Mathe-Sek-I-Ausbau von `M3` nach `M4`):

- **`canonical-math-sek1` / `canonical-math-sek2:`** `Mathematik (Gymnasium, DE)`, Motivationsanker `Warum Mathematik? - Entdecken, Muster & Alltag` bzw. `Warum Mathematik? - Denken, Muster & Zukunft`, terminale Prüfungsaufgaben unter `Prüfungen Jahrgangsstufe 5–10` bzw. `Übungen E-Phase`, `Q1`–`Q4` und `Übungen Prozesskompetenzen`.
- **`canonical-physics-sek1` / `canonical-physics-sek2:`** `Physik (Gymnasium, DE)`, Motivationsanker `Warum Physik?`, terminale Aufgaben unter `Übungen Sekundarstufe I Physik` bzw. `Übungen E-Phase` und `Q1`–`Q4`.

```
cd app
npm run quality:curriculum-status
npm run validate:graph
npm run validate:view-filters
npm run validate:composition-views
npm run quality:semantic-atomicity:check -- \
  --config=curricula/DE/Gymnasium/quality/semantic-atomicity/canonical-math-full.config.
json
```

## 3 Curriculum-Mapping-Workbench

Während das Qualitätsdashboard (2) anzeigt, *ob* QA-Regeln grün, gelb oder rot sind, macht die lokale schreibgeschützte **Curriculum-Mapping-Workbench** für Curriculum-Verantwortliche nachvollziehbar, *warum* ein Befund vorliegt — vom amtlichen Lehrplan-PDF bis in den lernendenseitigen SkillPilot-Baum.

```
Route:     /curriculum-mapping-workbench
Endpunkte: /__curriculum-mapping-workbench/list
           /__curriculum-mapping-workbench/load?sourceLandscapeId=<id>
```

### 3.1 Siebenteiliger Datenfluss im Repository

Die Workbench führt sieben bestehende Artefaktschichten ohne Schreibzugriff zusammen:

1. **Quelllandschafts-Register (`curricula/DE/Gymnasium/provenance/source-landscape-registry.json`):** Identifiziert Quell-Snapshots über `landscapeId`, `jurisdiction`, Titel und Quellpfad.
2. **Amtliche PDF-Passagenextraktion (`source-extraction/*.source-extraction.json`):** Für geprüfte Curricula (z. B. `DE_HE_MATHEMATIK_SEKII_KC2024.source-extraction.json` aus `kerncurriculum_gymnasiale_oberstufe-mathematik.pdf`) zerlegt `pdftotext` das PDF nach Themenfeldern (`E.1`, `Q1.3`, `Q4.2`), entfernt Kopf-/Fußzeilen, normalisiert Deutsch nach NFC gegen Mojibake (`ä`, `ö`, `ü`, `ß`) und wandelt Formeln in Inline-LaTeX für KaTeX um.
3. **Quellziele (Source Goals):** Stammen bei vorhandener `source-extraction` direkt aus den wörtlichen Teilaspekten der amtlichen Aufzählungspunkte (inkl. Eltern-Bullet, `sourceSpan` und PDF-Seite); fehlt eine persistierte Extraktion, fällt die Ansicht auf den Quell-Snapshot (`*.de.json.snapshot`) zurück.
4. **Source-Goal-Membership-Register (`source-goal-membership-registry.json`):** Definiert als erfasst geltende Quellziele; nicht registrierte Quellziele erscheinen rot.
5. **Source-Goal-Closure-Register (`source-goal-closure-registry.json`):** Zerlegt breite Originalziele in die sie abschließenden atomaren Quellziele.
6. **Mapping- und Review-Dateien (`curricula/DE/Gymnasium/mapping/...`):** Verknüpfen `legacyGoalId` bzw. extrahierte Quellziele (`exact`, `partial` oder explizite Lückenentscheidungen in `*.review.json`) mit `canonicalGoalId`.
7. **Kanonischer Skill-Graph und Kompositionsansicht:** Verbindet `curricula/DE/Gymnasium/canonical/*.de.json` mit der passenden lernendenseitigen Ansicht unter `curricula/DE/Gymnasium/composition-views/` (sodass rechts genau der für Lernende sichtbare Baum erscheint).

### 3.2 Fail-Closed-Pipeline-Gates (`MAPPING-1` bis `MAPPING-3`)

Ein späterer Mapping-Schritt darf niemals als abgeschlossen gemeldet werden, solange ein Vorgängerschritt offen oder blockiert ist (`pipelineStatus` in `source-extraction`):

| Gate | Bedeutung | Abschlusskriterium |
|---|---|---|
| `MAPPING-1` | Amtliche Passagen extrahiert | Jedes erwartete Themenfeld ist genau einmal mit amtlichem Aufzählungstext und sauberem Deutsch-/LaTeX-Anzeigetext vorhanden. |
| `MAPPING-2` | Quellziele gebildet | `MAPPING-1` komplett, jede zieltragende Passage hat mindestens ein eindeutiges Quellziel mit Span/Eltern-Bullet/Seitenreferenz und sauberem Encoding (reine Leitfaden-Hinweise dürfen zielfrei bleiben). |
| `MAPPING-3` | Kanonische Abbildung geprüft | `MAPPING-2` komplett, jedes akzeptierte Quellziel besitzt eine geprüfte Entscheidung auf existierende kanonische Ziele und ist inhaltlich abgedeckt (`needs_canonical_goal` oder `needs_view_placement_review` zeigen Fortschritt, schließen `MAPPING-3` aber nicht). |

### 3.3 Zweistufiges Interaktionsmodell und Prüfinterpretation

Die Oberfläche trennt die Prüfung in zwei zweigeteilte Stufen plus Detailleiste am unteren Rand:

- **Stufe 1 (Amtliche PDF-Passage ↔ Quellziele):** Links nach Themenfeldern segmentierte PDF-Passagen, rechts die zugeordneten Quellziele mit bidirektionaler Hervorhebung.
- **Stufe 2 (Quellziel-Baum ↔ SkillPilot-Baum):** Links der Quellziel-Baum mit Registrierungs- und Mapping-Status, rechts die lernendenseitige Kompositionsansicht mit direkter/abgeleiteter Quellenunterstützung und wechselseitigem Scroll-Fokus; unten Detailangaben zu PDF-Seite, Quellreferenz, Closure-Atomen und Zielknoten.

**Grün bedeutet registriert/verknüpft — nicht inhaltlich endabgenommen**

Ein grüner Status zeigt lediglich eine im Repository hinterlegte Registrierungs- oder Mapping-Beziehung. Die fachliche Prüffrage lautet stets, ob jedes Originalziel im PDF erfasst, jede breite Stelle sauber atomisiert, jedes Ziel semantisch passend zugeordnet und kein Bundesland-Ziel ohne legitimen Lehrplanbeleg eingeblendet ist.

Aktuell aufgelöste hessische Prüfeinstiege unter `http://127.0.0.1:5174/curriculum-mapping-workbench`:

| Fach | Stufe | Quellziele | Mappings | Amtliches PDF in der Workbench |
|---|---|---|---|---|
| Mathematik | Sek I | 132 | 37 | `g9-mathematik.pdf` |
| Mathematik | Sek II | 316 persistierte Ziele | 316 geprüft / 383 Kanten | `kerncurriculum_gymnasiale_oberstufe-mathematik.pdf` |
| Physik | Sek I | 60 | 53 | `g9-physik.pdf` |
| Physik | Sek II | 376 | 376 | `kerncurriculum_gymnasiale_oberstufe-physik.pdf` |

## 4 Curriculum-Paketbereitschaft (`full-standalone-v1`)

Dieses Gate beantwortet eine strengere Frage als der klassische Fachexport: Darf ein Artefakt ehrlich das eigenständige Laufzeitprofil `full-standalone-v1` beanspruchen? Reproduzierbare Fachexport-ZIPs bleiben nützliche Übergabe- und Roundtrip-Artefakte, beweisen aber keinen autarken Betrieb aus dem ZIP allein und erhalten daher explizit `not-ready-legacy` (`standaloneProfileReady: false`).

### 4.1 Statusvokabular und Exit-Codes

Die Richtlinie ist in `contracts/curriculum-package/v1/profiles/full-standalone-v1.readiness-policy.json` festgelegt; Berichte folgen `contracts/curriculum-package/v1/package-readiness-report.schema.json` (Evaluator `1.3.0` validiert seine eigene Ausgabe gegen dieses Schema):

| Status / Bedingung | Exit | Semantik |
|---|---|---|
| `ready` (oder exakter `--expect-status` / Selbsttest) | `0` | Alle blockierenden Prüfungen der versionierten Policy bestanden (für den Mathematik-Kandidaten verhindern offene Publikations-/Human-Gates diesen Status noch). |
| CLI-Aufruffehler | `2` | Ungültige Argumente oder fehlende Pflichtoptionen. |
| `invalid` | `10` | Eingabesicherheit, Zielidentität oder Vertragsbindung verletzt (ein unvollständiger Zielanspruch wird niemals zu Legacy herabgestuft). |
| `not-ready-incomplete` / `not-ready-legacy` | `20` | Zielvertrag konsistent deklariert, aber Vollständigkeits-, Publikations-/Redistributions- oder Consumer-Gates offen (`not-ready-incomplete`), bzw. reines Legacy-Fachexport-Artefakt (`not-ready-legacy`) ohne passenden `--expect-status`. |
| `unsupported` | `30` | Intern konsistenter Zielanspruch für eine von diesem Evaluator nicht unterstützte Vertrags-/Kompatibilitätsversion. |
| `--expect-status`-Abweichung | `40` | Ermittelter Status weicht vom explizit erwarteten Zielstatus ab. |
| Interner / Vertragsfehler | `70` | Interner Evaluatorfehler oder beschädigte Vertrauensverträge. |

### 4.2 Konformitätssuiten, Bau und hermetischer Paket-Consumer

Ausführung der Selbsttests und Vertragsprüfungen vom Repository-Root:

```
python3 -B scripts/validate_curriculum_package_contracts.py
python3 -B scripts/validate_curriculum_runtime_catalog_contract.py
python3 -B scripts/validate_curriculum_schema_catalog_contract.py
python3 -B scripts/evaluate_curriculum_package_readiness.py --self-test
python3 -B scripts/validate_full_standalone_curriculum_package.py --self-test
python3 -B scripts/run_package_consumer_smoke.py --self-test
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
python3 -B scripts/generate_curriculum_package_redistribution_review.py --check
python3 -B scripts/generate_curriculum_source_verification_review.py --check
```

Bau (`app/scripts/buildFullStandaloneCurriculumPackage.ts`), unabhängige ZIP-Validierung (`scripts/validate_full_standalone_curriculum_package.py`) und Readiness-Auswertung eines im Store provisionierten Staging-Kandidaten (bzw. komplett über `scripts/run_curriculum_full_package_conformance.sh`):

```
npm --prefix app run export:full-standalone-package -- \
  --release-root tmp/curriculum-release-model/mathematik-a \
  --output-dir tmp/curriculum-release-model/full-standalone-package \
  --zip --expect-entry-count 897 --expect-manifest-file-count 895 \
  --expect-binary-asset-count 715 \
  --expect-content-digest \
  sha256:9d3c0e03f95c06161abd0944335807704ffef1fcf20433f52d83167015637c64

PKG_DIR=tmp/curriculum-release-model/full-standalone-package
ZIP_NAME=skillpilot-curriculum-de-gymnasium-mathematik-0.1.0-conformance.3.json.zip
python3 -B scripts/evaluate_curriculum_package_readiness.py \
  --zip "$PKG_DIR/$ZIP_NAME" \
  --consumer-smoke-store tmp/curriculum-release-model/provisioned-store \
  --consumer-smoke-work-dir tmp/curriculum-release-model/package-consumer-smoke \
  --consumer-smoke-report "$PKG_DIR/package-consumer-smoke-report.json" \
  --report "$PKG_DIR/readiness-report.json" \
  --expect-status not-ready-incomplete
```

Der Evaluator ruft sowohl den unabhängigen Vollpaket-Validator v2 als auch den gepinnten Consumer-Runner (`contracts/curriculum-package/v1/package-consumer-smoke-report.schema.json`) selbst auf; ohne `--consumer-smoke-store` gilt ein externer Bericht lediglich als `external-unattested`. Der Consumer baut ein reines Runtime-Katalog-Frontend (`publicDir: false`) und ein schlankes Backend ohne eingebettete Curriculum-UUIDs oder statische Fachdaten und führt diese unter `bubblewrap --clearenv` bei ausgeblendetem Checkout, Loopback-Netzwerk, schreibgeschütztem Store, per `O_NOFOLLOW` gebundenem JDK/Python/Node/`strace` und realem Playwright-1.59.1-/Chromium-Lauf aus (inkl. 5 Poison-Sentinels, 15 geordneten Funktionschecks und Fail-Closed-UI-Marker bei Catalog-404 ohne Fallback).

### 4.3 Offline-Vertrauensmodell, Review-Trennung und Umsetzungsstand (`DPK-004a`–`DPK-008d`)

- **Offline-Auflösung und harte JSON-Grenzen:** Der paketlokale Schema-Katalog ist eine reine Hash-Auflösungstabelle unter der unterstützten Vertragsversion des Consumers (kein Remote-Abruf; max. `64 MiB` pro JSON-Eintrag, Verschachtelungstiefe `128`, max. `5.000.000` geparste Knoten). Auch der Runtime-Katalog löst Wurzeln, Scopes und Ansichten ausschließlich explizit auf.
- **Strukturelle Gültigkeit vs. Publikationsrecht:** Dateien mit `redistributionStatus: review-required` tragen zwingend `licenseExpression: null`; damit schlägt `publication.redistribution-cleared` fehl (`not-ready-incomplete`). `allowed` verlangt eine auf ein inventarisiertes Lizenzdokument aufgelöste Lizenzexpression (siehe 10 für die 715 Mathematik-Bilder, die 3 offenen Nicht-Binärklassen und die 479 offenen Quelltext-Entscheidungen).
- **Erreichte technische Meilensteine:** `DPK-004a/004b` und `DPK-005a/005b` etablieren Release-Modell, 22 gepinnte Payload-Schemata und deterministisches Streaming-ZIP32; `DPK-006a`–`006c` sichern den inhaltsadressierten Store samt CAS-Lock ohne Fallback; `DPK-007` beweist den 1,7-GB-Consumer-Pfad (15 Checks, 213 Frontier-Ziele). Der historische Freeze `DPK-007a` (`skillpilot-curriculum-de-gymnasium-mathematik-0.1.0-conformance.3.json.zip`, `914` Einträge, `757` Bilder, `1.738.161.217` Byte, SHA-256 `403cc0bc6004da549c8b9ed9fafad222fe0ddda1107806fe087cfa871a6dbcf9`, `contentDigest sha256:e83936aaf3645ff5f6e8132c4a801bd4bd66f55d3c0304a5deda3d6a5d194101`), der unabhängige

2,36-GB-FWU-OWL-Träger `DPK-008c` (`819` Einträge, `824.452` Tripel, SHA-256 `abab1d8a ac3e9394af26c614bbf231954ba45ab11f725dd0f93f088820dc3f94`, Placement-Schlüssel `goalId@unitId@index`) und die isolierte Rückübersetzung `DPK-008d` (`1.737.571.471` Byte, SHA-256 `7dcd233dd495900f6d6bd971ff6e86bdcdcfe5701f3522ed480be8336de23195`, 111/111 Normalform-Oracles, 20-teiliges Rawtrace-Evidenzinventar) bleiben unveränderliche historische Nachweise, ohne ein einziges menschliches Freigabegate vorwegzunehmen.

# 5 Curriculum-Release-Modell-Konformität

DPK-004 übersetzt den realen Mathematik-Authoring-Bestand des deutschen Gymnasiums in ein striktes, entpacktes Release-Modell (DPK-004a für die Laufzeitanteile, DPK-004b für die maßgebliche Publikationsevidenz) als ausführbare Brücke zum späteren full-standalone-v1-JSON-ZIP (4). DPK-004 bis DPK-007a sind technisch bestanden (passed).

## 5.1 Überblick und vertrauenswürdige Eingaben

Der Compiler leitet keine Semantik aus Dateinamen oder Zieltiteln ab, sondern bindet vier explizite Eingaben:

1. **Release-Modell-Build-Profil** (contracts/curriculum-package/v1/profiles/de-gymnasium-mathematik-release-model-v1.profile.json): Quell-/Ausgabepfade, Paketidentität, Views, Decks, Ressourcenregeln, Pfadrelokationen, Verträge und exakte Zähler (1 Landschaft, 1.147 Ziele, 88 Views, 12 Decks, 128 Karten, 784 Ressourcenlinks, 715 aktive Bilder mit 1.549.052.741 Byte).
2. **Ontologie-Profil** (contracts/curriculum-package/v1/profiles/de-gymnasium-mathematik-v1.profile.json, Version 1.1.1): Durch SHA-256 gepinnt; bindet das Core-first-Mapping an Repository, Checkout-Pfad (tmp/lehrplan-ontologie), Commit, src/ontology/components/lehrplan-core.owl, Ontologie-IRI und Datei-Digest.
3. **Semantisches Zielarten-Ledger** (curricula/DE/Gymnasium/quality/release-model/mathematik.semantic-kinds.json): Aktueller Quell-Fingerprint für jedes der 1.147 Ziele ohne Titel-/ID-Heuristiken.
4. **Feldsemantik-Registry** (contracts/curriculum-package/v1/profiles/skillpilot-fwu-field-semantics-v1.registry.json): 456 explizite Einträge (325 Runtime, 130 Publikation, 1 generischer Bestandseintrag; SHA-256 d19497f0b18b0c7733dcaf6f5dc63590352814c2d764c858b333751b5af25e9d), Normalisierungsprofil und geschlossene Draft-2020-12-Schemata (bidirektional ohne tote Registry-Pfade geprüft; field-coverage.json weist konkret belegte vs. reine Schemafelder getrennt aus).

| Semantische Zielart (semantic Kind) | Anzahl | Rolle im Mathematik-Release-Modell |
|---|---|---|
| curricularAtomic | 780 | Fachliche atomare Kompetenzziele (711 mit aktivem Bild, 69 providerbedingt zurückgestellt, 152 menschlich freigegeben) |
| curricularArea | 218 | Fachliche Themen- und Bereichscluster |
| practiceAssessment | 129 | Übungs-, Klausur- und Abitur-Aufgabenknoten |
| programStructure | 7 | Programmstrukturelle Container |
| memory | 6 | SRS-/Merkwissen-Ziele |
| runtimeSupport | 5 | Technische Laufzeit-Stützknoten |
| orientation | 2 | Orientierungs- und Motivationsanker |

## 5.2 Erzeugte Modellstruktur, Digests und Publikationsevidenz

Die Ausgabe entsteht atomar per Staging-Verzeichnis unterhalb von tmp/ (conformance-model-only-not-a-package, Symlinks werden abgewiesen; Bildbytes selbst werden erst im Begleit-ZIP materialisiert):

```
data/
  canonical/mathematik.landscape.json
  views/index.json & views/*.json
  cards/card-index.json & cards/*.json
  resources/resource-index.json
  mappings/source-to-canonical.json
  sources/source-index.json & sources/source-goal-references.json
  runtime/{catalog,dependency-closure,migration-aliases}.json
  assessment-sources/...
metadata/
  quality/release-quality-evidence.json
  {semantic-content-index,field-coverage,build-inputs,release-model-conformance}.json
```

- **Dependency-Closure und Digests:** `dependency-closure.json` umfasst `2.429` Runtime-Definitionen und `20.523` harte Referenzen ohne externe Abhängigkeiten (`sha256:5d542bfb46658aca0f570fa21d4771008d3238ab4aa87e5a551d55ed9efc3bc6`; Definitionsindex `sha256:1af9cf9fecdcdfec27372c1fc57a700634efb998755070ffd84dff0d1cff0680`). `migration-aliases.json` nutzt für das Erstrelease eine `initial`-Baseline. `semantic-content-index.json` bündelt `111` logische Artefakte und `715` Bildrecords pfadneutral in `contentDigest sha256:9d3c0e03f95c06161abd0944335807704ffef1fcf20433f52d83167015637c64` (während der historische Freeze `.3` aus `DPK-007a/008` unverändert erhalten bleibt). Ist ein Voraussetzungs-Reduktionsregister deklariert, prüfen Compiler und Validator unabhängig den Node-Beweis (`.nvmrc`) vor dem Fingerprint-Abgleich.
- **Vier Publikationsrollen (`runtimeRequired: false`):** `mapping` (`source-to-canonical-mappings`: `10.021` Entscheidungen → `33.234` exakte/partielle Kanten ohne unsichere `missing => exact`-Annahme), `source-index` (`official-source-index`: `31` Sammlungen, `55` Dokumente — `52` Core-`Lehrplan` + `3` `OfficialSourceDocument` — in `16` Ländern), `source-goal-reference-index` (`9.977` geprüfte Quellziele als Core `Curriculares Element` mit `CE-Verweis`; C0-Steuerzeichen und ungepaarte UTF-16-Surrogate scheitern fail-closed) und `quality-evidence` (`release-quality-evidence`: `2.335` Entscheidungen aus je `780` Atomizitäts-/Memory-Zielen, `64` aktiven Karten und `711` aktiven Bildern; `publication Status: not-ready`).

### 5.3 Prüfkommandos und gehosteter PDF-Modus

Kompilierung, unabhängige 49-Fälle-Validierung, Fixture-Tests (9 positiv, 22 negativ, ohne Remote-Schemaabruf) und der verpflichtende JSON-Konformitätswrapper (ohne 1,7-GB-ZIP):

```
PROFILE=contracts/curriculum-package/v1/profiles
PROFILE="$PROFILE/de-gymnasium-mathematik-release-model-v1.profile.json"
python3 -B scripts/compile_curriculum_release_model.py \
  --profile "$PROFILE" \
  --output tmp/curriculum-release-model/mathematik-a
python3 -B scripts/validate_curriculum_release_model.py \
  --profile "$PROFILE" \
  --release-root tmp/curriculum-release-model/mathematik-a
python3 -B scripts/validate_curriculum_release_model_fixtures.py
python3 -B scripts/compile_curriculum_release_model.py \
  --self-test-dependency-emission
bash scripts/run_curriculum_release_model_conformance.sh
```

Lokal müssen referenzierte Quell-PDFs reguläre Dateien (keine Symlinks) sein und werden byte-genau gehasht. Da gehostete GitHub-Checkouts die per Git ignorierten amtlichen PDFs nicht enthalten, setzen die Workflows explizit `SKILLPILOT_SOURCE_PDF_MODE=committed-bindings`:

Nur tatsächlich fehlende PDFs dürfen auf Bytezahl und SHA-256 des eingecheckten Source-Verification-Ledgers zurückfallen, während vorhandene, geänderte oder per Symlink verlinkte Dateien weiterhin strikt geprüft werden.

# 6 Duale Curriculum-Release-Verträge

`DPK-003` legt den Fail-Closed-Semantikvertrag für ein gekoppeltes Paar aus SkillPilot-JSON-Laufzeitpaket und Core-first-FWU-Ontologiepaket fest, ohne damit bereits ein Produktivpaket zu veröffentlichen.

## 6.1 Verbindliche Vertragsinvarianten

- **Lückenlose Feld-Registry:** Jedes Feld des Release-Modells muss exakt einem versionierten Registry-Eintrag entsprechen (Skalare, geordnete Listen, Mengen, Maps, Binärreferenzen, generierte nicht-semantische Felder). Unbekannte Felder blockieren das Ontologie-Release.
- **Explizite nullbasierte Ordnung in RDF:** Lernziele, direkte `contains`- und `requires`-Kanten, Ressourcenlinks, View-Kinder, Karten und Bewertungsschritte sind geordnet. RDF speichert eindeutige, lückenlose nullbasierte Positionen, da ungeordnete Core-Relationen die Autorenreihenfolge nicht rekonstruieren können.
- **Strikte Wertunterscheidung & Verbot breiter JSON-Container:** Fehlend, `null`, leer und Standardwert bleiben getrennt, sofern ein einzelner Registry-Eintrag nichts anderes festlegt. Bekannte curriculare Konzepte nutzen den FWU-Core; SkillPilot-Vokabular dient nur der Laufzeitsemantik und exakten Rückübersetzung. Kanonische JSON-Literale erfordern eine feldspezifische Core-Lücke, Granularität und Byte-Obergrenze; paket- oder landschaftsweite JSON-Träger sind verboten.
- **Pfadneutrale Binäridentität:** Binärressourcen gehen über stabile Ressourcen-ID, kanonische öffentliche Referenz, MIME-Typ, Byte-Länge und SHA-256 in den gemeinsamen Index ein; variantenspezifische ZIP-Pfade ändern `contentDigest` nicht.
- **Äquivalenzbericht & Signaturprojektion:** Ein bestandener Äquivalenzbericht bindet beide finalen ZIPs und Manifeste, Verträge, FWU-Core-IRI, Toolchain, Feldabdeckung, Rollen-Digests, Graph-/View-/Karten-/Asset-Vergleiche, isolierte Rückübersetzung, Validierung beider JSON-Pakete, hermetische Consumer-Läufe und Reproduzierbarkeit. Der externe Release-Index entfernt vor dem kanonischen Hashing den Signaturblock (keine Selbstreferenz); der `DPK-003`-Validator weist jeden `verified`- oder `stable`-Anspruch ab, bis `DPK-011` Signatur-, Identitäts- und Provenienzprüfungen ausführt (`staging` darf unsigniert bleiben).

## 6.2 Ausführbares Gate und Abgrenzung

Ausführung vom Repository-Root (Teil der optionalen Ontologie-Lane in `./run_ci.sh owl`, `./run_ci.sh full` und `.github/workflows/owl-ci.yml`; erfordert `jsonschema==4.26.0`):

```
python3 -B scripts/validate_curriculum_dual_release_contracts.py
```

Der Validator weist doppelte JSON-Schlüssel, nicht-endliche Zahlen, ungültiges Unicode und RDF/XML-unsichere Steuerzeichen ab, prüft alle Schemata und Positiv-Fixtures, berechnet logische/binäre Records sowie den Gesamt-`contentDigest` neu und führt eine Mutationsmatrix aus (verlorene Ordnung, mehrdeutige Pfade, verwischte `missing`/`null`-Semantik, unzulässige JSON-Träger, gefälschte Digests, substituierte Berichte, nicht-isolierte Rekonstruktion, unbestätigte `stable`-Promotion).

**Synthetische Vertrags-Fixtures sind keine Curriculum-Evidenz**

Die positiven Release- und Äquivalenzdateien unter `contracts/curriculum-package/v1/fixtures/dual-release/valid/` enthalten synthetische Hashes und Zähler ausschließlich

zum Vertragstest und dürfen niemals als Curriculum-Nachweis veröffentlicht werden.

## 7 Prüfung der Requires-Relationen

Dieses Kapitel dient als stabiler Kompatibilitäts-Einstiegspunkt für Verweise auf die strukturellen Relationsprüfungen.

Der vollständig gepflegte Regelkatalog befindet sich in Graph-Validierungsregeln (`GVR`/`APV`/`CPV`) in Band IV des Qualitätshandbuchs. Maßgeblich dort dokumentiert sind:

- sämtliche implementierten Validatorregeln (`GVR-*`, `APV-*`, `CPV-*`),
- das strikte Standard-Durchsetzungsprofil in der CI (`npm run validate:graph`),
- der lokale Warnmodus für Altbestands-Analysen (`VALIDATE_GRAPH_STRICT_RULES=0 npm run validate:graph`).

Für das inhaltlich-didaktische Review einzelner Voraussetzungskanten siehe ergänzend den manuellen Voraussetzungs-Reviewprozess in 9.

# 8 Atomizitäts-Reviewprozess

Dieses Kapitel beschreibt den wiederholbaren manuellen Autoren- und QA-Workflow, mit dem bestehende Blattziele (`contains: []`) einer Skill-Landschaft auf echte didaktische Atomizität geprüft und gebündelte Ziele sauber aufgeteilt werden (ergänzend zum maschinell gestützten Ledger-Gate `CQR-301` in Semantische Atomizität).

## 8.1 Geltungsbereich und Arbeitsdefinition

Geprüft wird jeweils eine Landschaftsdatei pro Durchgang über alle Ziele mit `contains: []`. Bewusst **ausgenommen** sind Zieltypen mit eigenen QA-Lanes:

- Memorierungsziele mit `memorization` oder `srs-deck:*`,
- Prüfungs- und Aufgabenbündel mit `examData`,
- Motivations- und Orientierungsziele (`Motivation`, `Orientation`).

Ein fachliches Lernziel gilt als **atomar**, wenn alle vier Kriterien erfüllt sind:

1. Es beschreibt genau **eine** tragende Kompetenzhandlung.
2. Es ist realistisch in **1–3 Aufgaben** überprüfbar.
3. Es bildet relativ zu seinen Voraussetzungen **einen kleinen didaktischen Lernschritt**.
4. Es versteckt keine unabhängigen Teilkompetenzen hinter einem Sammelsatz (Warnsignale: mehrere Verben unterschiedlicher kognitiver Stufen, mehrere Teilgebiete oder getrennte Nachweispflichten).

## 8.2 Siebenstufiger Review-Ablauf und Entscheidungsdatei

Das Werkzeug `app/scripts/generateAtomicReport.ts` (`npm run report:atomic`) bereitet Entscheidungsgerüst und Befundbericht (`../tmp/atomic_findings_<curriculum>.md` mit Zusammenfassung, `not_ok`-Befunden und Komplettanhang) vor:

```
cd app
npm run report:atomic -- \
  --input ../curricula/<...>/json/<file>.json \
  --decisions ../tmp/atomic_decisions_<name>.json \
  --init-decisions
```

1. Entscheidungsgerüst initialisieren oder aktualisieren.
2. Jedes Blattziel semantisch zerlegen (Handlungsverben, Gegenstandsbereich, erwarteter Kompetenznachweis).
3. Manuell prüfen, ob eine Aufteilung in 2–4 Teilziele Klarheit und Prüfbarkeit substanziell verbessert (nur bei echtem didaktischem Präzisionsgewinn `not_ok` vergeben, nicht aus rein stilistischen Gründen).
4. Status (`ok`, `not_ok`, `pending`) in der JSON-Entscheidungsdatei festhalten:

```
{
  "goals: {
    «goalId-a>: { ßtatus: ök" },
    «goalId-b>: {
      ßtatus: "not_ok",
      "problem: "warum das Ziel gebuendelt ist",
      "proposal: "wie es in 2-4 Teilziele geschnitten wird"
```

```
    },
    «goalId-c»: { ßtatus: "pending" }
  }
}
```

1. Für `not_ok`-Ziele einen konkreten Schnitt ausarbeiten (`title`, `description`, `requires`, Eltern-`contains`).
2. Nach Umsetzung Zyklenfreiheit, Referenzintegrität und Gewichte prüfen (atomare Ziele behalten in der Regel `weight = 1`; Clustergewichte spiegeln die Anzahl eindeutiger atomarer Nachfahren wider).
3. Abschließend `cd app && npm run validate:graph` ausführen.

# 9 Voraussetzungs-Reviewprozess (`requires`)

Dieses Kapitel dokumentiert den wiederholbaren manuellen Autoren- und QA-Workflow zur Prüfung der Voraussetzungsqualität auf atomaren Lernzielen (`contains: []`): Ist der didaktische Lernsprung von den Voraussetzungen zum Ziel klein genug, und werden grobe oder irreführende Kanten durch präzise atomare Routen ersetzt?

## 9.1 Übergangssemantik vs. kanonischer Zielzustand

Das Report-Werkzeug `app/scripts/generateRequiresReport.ts` (`npm run report:requires`) macht beide Schichten des aktuellen Kompatibilitätsmodells mitsamt Herkunft sichtbar:

- **`direct requires`:** Unmittelbar am Atomziel selbst hinterlegte Voraussetzungen.
- **`effective requires`:** Direkte Voraussetzungen zuzüglich aller von Vorgänger-Clustern über `contains` geerbten Voraussetzungen.

Während diese Sicht für die Bestandsmigration unverzichtbar ist, bleibt der verbindliche Zielzustand eine rein atomare Voraussetzungsschicht $R \subseteq A \times A$ (Graph-Definition).

## 9.2 Sechsstufiger Prüf- und Reparaturablauf

```
cd app
npm run report:requires -- \
  --input ../curricula/<...>/json/<file>.json \
  --decisions ../tmp/requires_decisions_<name>.json \
  --init-decisions
```

1. Entscheidungsgerüst erzeugen oder aktualisieren (Standardbericht: `../tmp/requires_findings_<curriculum>.md` mit Zusammenfassung, `not_ok`-Liste und Komplettanhang für `direct requires`, `effective requires` und Ursprungscluster/-ziele).
2. Für jedes Atomziel direkte und effektive Voraussetzungen samt Vererbungsursprung sichten.
3. Das **didaktische Delta** formulieren: Was kann die lernende Person nach den Voraussetzungen bereits, was kommt im Ziel neu hinzu, und ist dieser Zuwachs genau ein lehrbarer Schritt?
4. Entscheidung (`ok`, `not_ok`, `pending`) in der JSON-Datei erfassen (`not_ok` nur bei echtem didaktischem Problem wie zu großem Lernsprung, fachfremder Blockade oder verdeckter Cluster-Abhängigkeit — nicht für bloße Graph-Kosmetik):

```
{
  "goals: {
    «goalId-a»: { ßtatus: ök" },
    «goalId-b»: {
      ßtatus: "not_ok",
      "problem: "was didaktisch nicht traegt",
      "proposal: "wie die requires-Struktur repariert wird"
    },
    «goalId-c»: { ßtatus: "pending" }
  }
}
```

1. Bei `not_ok` eine der vier Standardreparaturen anwenden: (a) Brückenziel ergänzen, (b) Zielknoten aufteilen (8), (c) Cluster-`requires` durch konkrete atomare Voraussetzungen ersetzen, oder (d) redundante bzw. überblockierende Kanten entfernen.
2. Nach den Änderungen `cd app && npm run validate:graph` ausführen.

## 10 Menschliche Review-Gates für Curriculum-Releases (HR-001–HR-006)

Dieses Kapitel bündelt die sechs strikt voneinander unabhängigen menschlichen Freigabegates, die keine Toolchain automatisch entscheiden darf (zunächst bezogen auf `org.skillpilot.curriculum.de.gymnasium.mathematik`; jede Hash- oder Inhaltsänderung setzt betroffene Freigaben auf stale zurück).

### 10.1 Überblick der sechs unabhängigen Gates

| Gate | Offener Umfang | Blockiert | Primäre Arbeitsquelle |
|---|---|---|---|
| `HR-001` Bildqualität | `559` aktive Bilder offen (`152` freigegeben; `69` von `780` Zielen providerbedingt zurückgestellt) | Fachliche Publikationsreife | `curricula/DE/Gymnasium/quality/goal-visualization-qa/mathematik.qa.json` |
| `HR-002` Bildrechte | `715` Einzelentscheidungen (`708` Provider-Pipeline, `6` Nutzer-Claim, `1` deterministischer SVG-Render) | Öffentliche Weiterverbreitung | `curricula/DE/Gymnasium/quality/package-redistribution/de-gymnasium-mathematik-v1.review.json` |
| `HR-003` Nicht-Binärrechte | `3` Klassenentscheidungen (`skillpilot-data`, `official-source-evidence`, `generated-metadata`) | Öffentliche Weiterverbreitung | Dasselbe Redistribution-Ledger |
| `HR-004` Quelltext-Verifikation | `479` Einzelentscheidungen (neben `9.498` technischen Treffern) | Quellen-QA / Promotion | `curricula/DE/Gymnasium/quality/source-verification/de-gymnasium-mathematik-v1.review.json` |
| `HR-005` Ontologie-Sign-off | `1` hashgebundene fachlich-ontologische und rechtliche Expertenfreigabe | Ontologie-Promotion | `DPK-008c`-FWU-Receipt & `DPK-008d`-Reverse-Receipt |
| `HR-006` Signatur & `stable` | `1` Trust-/Release-Entscheidung je Releasegruppe | Öffentlicher `stable`-Release | Künftiger `DPK-011`-Release-Index |

### 10.2 Fach-, Rechte- und Quelltext-Gates (HR-001 bis HR-004)

- **`HR-001` (Fachliche Bildfreigabe):** Prüft mathematische Korrektheit aller Werte/Relationen, Passung zu Lernziel und Altersstufe sowie Lesbarkeit, Sprache und Alt-Text auf dem aktuellen Asset-Hash (`human-approved`, `rejected`, `deferred-provider-limitation`).
- **`HR-002` (Rechte an 715 Bildern):** Alle eingebetteten Bilder stehen zunächst auf `pending-human-review`, `review-required` und `licenseExpression: null` (`708` Provider-Pipeline-Claims, `6` `user-provided-generated-claim` mit zusätzlicher Uploader-Attestation, `1` deterministischer Render via `librsvg rsvg-convert` 2.52.5 aus reviewtem SkillPilot-SVG). Provenienzlabels sind keine Lizenz; nötig sind Provider-/Lizenzprüfung, SPDX-artige `licenseExpression`, Evidenz, Reviewer und Zeitstempel auf allen 715 Records (nicht belegbare Assets werden `prohibited` und vor Publikation entfernt/ersetzt).
- **`HR-003` (Rechte der Nicht-Binärartefakte):** Nur `software-contracts` (gebunden an den Root-`LICENSE`-Hash) ist automatisch Apache-2.0. Offen sind (1) `skillpilot-data`, (2) `official-source-evidence` (Trennung zwischen Fakten/Metadaten, eigener Paraphrase und geschütztem Wortlaut) und (3) `generated-metadata`.

- **HR-004 (479 Source-Text-Entscheidungen):** Neben 9.493 Treffern in authored Passage-Carriern und 5 temporären PDF-Projektionstreffern (9.498 technische Nachweise ohne Rechtsfreigabe) erfordern 479 Records eine durch reviewEvidenceSha256 an sourceText, Passage, PDF und Locator gebundene Entscheidung (verified-verbatim, verified-normalized-transcription, reviewed-paraphrase, rejected; Aufschlüsselung in source-verification-de-gymnasium-mathematik-v1.md).

### 10.3 Ontologie-Sign-off und Stable-Promotion (HR-005 & HR-006)

- **HR-005 (Core-first Ontologie-Sign-off):** Hashgebundene Expertenfreigabe über das DPK-008c-FWU-OWL-ZIP (2.362.455.128 Byte, SHA-256 abab1d8aac3e9394af26c614bbf231954ba45ab11f725dd0f93f088820dc3f94, Manifest 29f308424d1aeba9095f0e800253acadcdfaca0562dfa1fc37741c77c76023b3), das rekonstruierte JSON-ZIP (1.737.571.471 Byte, SHA-256 7dcd233dd495900f6d6bd971ff6e86bdcdcfe5701f3522ed480be8336de23195, Manifest 8d7970435431ff78743d0bf413a54bbebee009bbaafe35f1283ebcb89b4f2ff0), das DPK-008d-Reverse-Receipt (12.185 Byte, 94035f2d88e2c2b7588f13f01cb44b917c83900b9f839c66558ea7eae585884f), das 20-teilige Evidenzmanifest (3.483 Byte, 31c256ff612aa7a346fb0792c83b109d4bc01df52fdc36e98a8c717bf9b8eafc, Inventardigest 8483f5ce428cc54cca1353668370ae09f32f51166af6a3235a2ff19ccb88ea1e) und die 454-Einträge-Registry (2e536c3f8d63e2acf45690375ace69ec0c6a6e92787bc8a16957b80120c4ca48). Zu bestätigen sind semantische Passung jedes FWU-Core-Terms, Begrenzung von Erweiterungen und JSON-Literalen, saubere Schichtentrennung sowie Lizenz-/Share-Alike-Pflichten (CC-BY-SA-4.0; bis zum Sign-off bleibt die Core-Datei im Manifest review-required mit licenseExpression: null).
- **HR-006 (Signatur und stable-Promotion nach DPK-011):** Bestätigt die vertrauenswürdige Signing-Identität und den Abschluss aller vorherigen Human-Gates auf den gebundenen Hashes vor der atomaren Promotion der JSON-/OWL-Releasegruppe von staging auf stable.

## 11 MEM-SPARQL-Konsistenzaudit

Diese bewusst **nicht-blockierende** QA-Lane vergleicht Live-Daten des externen MEM/FWU-Lehrplan-Ontologie-Endpunkts (`https://sparql.mem.edufeed.org/sparql`) mit den persistierten `source-extraction`-Nachweisen in SkillPilot. Da die Länderabdeckung im externen Endpunkt noch unvollständig ist, dienen Abweichungen als strukturierte Triage-Signale und nicht als CI-Blocker; MEM ersetzt weder amtliche Quellenreferenzen noch `CQR-*`-, Atomizitäts-, Merkkarten- oder View-Validierungen.

### 11.1 Maßgebliche Quellen, generierte Artefakte und Ausführung

- **Konfiguration & Ledger (Source of Truth):** `curricula/DE/Gymnasium/quality/mem-sparql-consistency/canonical-math-poc.config.json` und `canonical-math-poc.review.jsonl` (Generator: `app/scripts/generateMemSparqlConsistencyAudit.ts`).
- **Generierte Snapshots (nicht manuell editieren):** mem-sparql-consistency-audit.md, `mem-sparql-consistency-audit.json`, mem-sparql-consistency-review-issues.md und `mem-sparql-consistency-review-issues.json`.

```
cd app
npm run quality:mem-sparql-consistency
npm run quality:mem-sparql-consistency -- \
  --endpoint=https://sparql.mem.edufeed.org/sparql
MEM_CFG=curricula/DE/Gymnasium/quality/mem-sparql-consistency
npm run quality:mem-sparql-consistency -- \
  --config="$MEM_CFG/canonical-math-poc.config.json"
```

Der Befehl endet auch bei offenen Triage-Punkten mit Exit-Code `0` und schlägt nur bei lokalen Skript- oder Konfigurationsfehlern fehl (Initialuntersuchung siehe MEM-SPARQL-PoC vom 01.06.2026; operative Checkliste siehe 12).

### 11.2 Fingerprint-gebundenes Review-Ledger und Erweiterungskriterien

Jeder Eintrag der JSON-Review-Queue trägt einen aus der aktuellen MEM-/Lokal-Evidenz berechneten `itemFingerprint`:

```
{
  ßchemaVersion: 1,
  äuditId: "canonical-math-poc",
  itemId: "triage-a3e85d9ef77a",
  itemFingerprint: ßha256:...",
  ßtatus: "mem_feedback_candidate",
  "reviewedAt: "2026-06-01",
  "reviewer: initials-or-tool",
  "reason: SShort evidence-based decision.",
  "followUpRef: öptional issue, PR, or MEM feedback reference"
}
```

- **Zulässige `status`-Werte:** `not_an_issue`, `local_extraction_fix_needed`, `mem_feedback_candidate`, `source_version_gap`, `deferred`.
- **Ledger-Validierung bei jedem Lauf:** Passende `itemId` + `itemFingerprint` markieren den Eintrag als `decided` (`deferred` bleibt offen); geänderte Fingerprints erscheinen als `stale_review`; unbekannte, doppelte oder fehlerhafte Zeilen landen unter `reviewLedger.diagnostics`.

- **Aufnahmekriterien für neue Vergleiche:** Nur ergänzen, wenn MEM konkrete Plandaten (nicht nur Vokabular) bereitstellt, Graph- und Klassen-IRIs bekannt sind, ein lokales `source-extraction`-Artefakt für denselben Scope vorliegt und stabile Issue-IDs ohne Blockade bestehender `CQR-*`-Regeln erzeugt werden können.

## 12 MEM-SPARQL-Konsistenz-Runbook

Dieses Runbook bildet die operative Checkliste zur Ausführung und Triage des in 11 spezifizierten MEM-SPARQL-Konsistenzaudits (Erstprotokoll siehe MEM-SPARQL-PoC vom 01.06.2026).

### 12.1 Audit ausführen und Queue inspizieren

```
cd app
npm run quality:mem-sparql-consistency
cd ..
node - <<'NODE'
const q = require('./docs/qa-ci/status/mem-sparql-consistency-review-issues.json')
console.log(q.summary)
console.log(q.reviewLedger)
console.log(q.highestSignalChecks.map((i) => ({
  id: i.id, status: i.status, category: i.category,
  side: i.side, mem: i.memText, local: i.localText,
})))
NODE
```

Zur menschlichen Durchsicht dienen mem-sparql-consistency-review-issues.md (Queue) und mem-sparql-consistency-audit.md (Länderabdeckung und Zähler).

### 12.2 Entscheidungen eintragen, Diagnostik deuten und lokal verifizieren

1. Eintrag aus `highestSignalChecks` oder `textTriageQueue` wählen und `id` sowie `itemFingerprint` übernehmen.
2. `memText`/`memRef`, `localText`/`localRef`, `rationale` und ggf. `reviewNote` abgleichen; lokale Extraktionsfehler direkt im `source-extraction`-Artefakt beheben.
3. Status wählen (`not_an_issue`, `local_extraction_fix_needed`, `mem_feedback_candidate`, `source_version_gap`, `deferred`), als einzeiliges JSONL-Objekt an `curricula/DE/Gymnasium/quality/mem-sparql-consistency/canonical-math-poc.review.jsonl` anhängen und das Audit erneut starten.

Bedeutung von `reviewLedger.diagnostics` (vor einem MEM-Feedback-Durchgang bereinigen; Testdaten nur unter `tmp/` anlegen):

- `stale_review`: Eintrag existiert weiterhin, aber der Evidenz-Fingerprint hat sich geändert → neu prüfen.
- `unknownItemRecords`: Ledger verweist auf eine nicht mehr in der aktuellen Queue enthaltene `itemId`.
- `duplicateItemRecords`: Mehrere Zeilen zur selben `itemId` (die letzte gewinnt; Mehrdeutigkeit entfernen).

Lokale Verifikation nach Änderungen an Generator, Konfiguration, Extraktion oder Dokumentation:

```
cd app
npx eslint scripts/generateMemSparqlConsistencyAudit.ts
npm run quality:source-coverage-audit:check
cd ..
git diff --check
MEM_CFG=curricula/DE/Gymnasium/quality/mem-sparql-consistency
```

```
node -e "for (const f of [
  '$MEM_CFG/canonical-math-poc.config.json',
  'docs/qa-ci/status/mem-sparql-consistency-audit.json',
  'docs/qa-ci/status/mem-sparql-consistency-review-issues.json'
]) JSON.parse(require('fs').readFileSync(f, 'utf8')); console.log('json ok')"
```

## 13 Runbook für Lernziel-Quellenbegründungen

Dieses Runbook beschreibt Erzeugung, Laufzeitbereitstellung und Prüfung menschenlesbarer Quellenbegründungen (`generateGoalSourceRationales.ts`) entlang der geprüften klassischen Kette `canonical goal -> mapping review -> source extraction -> official source document` (Fachkonzept siehe `docs/concept/skill-graph/human-readable-source-rationales.md`, Statusverzeichnis status/README.md).

### 13.1 PoC-, Zielmengen- und MEM/FWU-Showcase-Kommandos

Ausführung aus `app/` für den Bayern-Mathematik-PoC (goal-source-rationales-poc.md + `.json`), benutzerdefinierte Zielmengen und die Bayern-LehrplanPLUS-Showcases mit Live-MEM/FWU-Abgleich (`:plain` blendet Ziel-UUIDs zugunsten des SkillPilot-Pfads aus und ergänzt konkrete SPARQL-Abfragen):

```
cd app
npm run quality:goal-source-rationales:poc
npx tsx scripts/generateGoalSourceRationales.ts \
  --jurisdiction=DE-BY --goals=<goal-id>,<another-goal-id>
npm run quality:goal-source-rationales:mem-examples
npm run quality:goal-source-rationales:mem-examples:plain
```

Wichtige CLI-Optionen: `--landscape=<path>`, `--mapping-root=<path>`, `--output-json=<path>`, `--output-md=<path>`, `--no-md`, `--goals=source-backed | source-backed-relevant-leaves | all-relevant-leaves`, `--jurisdiction=all`, `--include-mem` und `--mem-config=<path>`.

### 13.2 Öffentlicher Laufzeitindex (Repository- vs. Paketmodus) & Deployment-Smoke

- **Repository-Modus:** Der Explorer lädt bei Bedarf genau zwei Kompatibilitätsindizes außerhalb von JS-Bundle und Service-Worker-Precache nach: `/data/goal-source-rationales-math-public.json` (bevorzugt `DE-BY` für sichtbaren MEM/FWU-Abgleich) und `/data/goal-source-rationales-physics-public.json` (die generische Deck-Route darf sie nicht abfangen).
- **Paketmodus:** `/data/**` ist gesperrt; das UI ermittelt die Bindungen über Catalog API 1.2 (`/api/ui/curriculum-catalog`) und lädt Einzelziele generationsgebunden über `/api/ui/curriculum-source-evidence/packages/{packageId}/{packageVersion}/goals/{goalId}` ohne Repository-/Classpath-Fallback.

Erzeugung, Build-Artefaktprüfung und Live-Deployment-Smoke (automatisch im Deploy-Skript nach Service-Neustart ausgeführt; erkennt anhand von `/api/ui/curriculum-catalog` `404` vs. `1.2` den aktiven Modus):

```
cd app
npm run quality:goal-source-rationales:public
npm run check:goal-source-rationales:math-public
npm run check:goal-source-rationales:build-artifact
npm run smoke:goal-source-rationales:deployment
npm run smoke:goal-source-rationales:deployment -- --base-url=http://127.0.0.1:8080
cd ..
SKILLPILOT_BASE_URL=https://staging.example.org \
  ./deploy_skillpilot.sh --coach-variant openai-mcp
```

### 13.3 Review-Checkliste und Folge-Issues

Je gerendertem Ziel sind fünf Punkte zu prüfen:

1. `sourceRationaleStatus:` `classic_source_reviewed` (regulär zulässig), `classic_source_partial` (nur mit nachvollziehbarer Teil-Begründung) oder `classic_source_gap` (erfordert nicht-blockierendes Folge-Issue).
2. **Originalquelle finden:** Offizielle URL oder archivierter Quellpfad samt konkretem Such-/Abschnittshinweis vorhanden.
3. **Extrahierter Quellenbeleg:** Dokumentmetadaten aufgelöst, `sourceGoalId` und Referenz konsistent, Markdown-Auszug knapp gehalten (Volltext nur im JSON).
4. **Begründung des SkillPilot-Ziels:** Verständliche Zuordnungsbegründung ohne Übertreibung von `partial`-, `split`- oder `aggregate`-Mappings.
5. **MEM/FWU-SPARQL-Route:** `mem_sparql_unavailable` im klassischen PoC bzw. `mem_sparql_consistent` im Bayern-Showcase (stets sekundäre Route, solange kein Scope als Primärquelle freigegeben ist; bei Widersprüchen oder fehlenden Metadaten manuelles Review-Issue anlegen).

# 14 Abi-2026 Mathe DeepLink MVP (BlueSky)

Dieses Kapitel dokumentiert die Linkstruktur, automatische Cockpit-Vorkonfiguration, Kampagnen-Telemetrie und QA-Prüflinks des DeepLink-MVPs für das hessische Mathematik-Abitur 2026.

## 14.1 Linkstruktur und Cockpit-Vorkonfiguration

Die Zielroute `/start/abi26-he-mathe-k1` akzeptiert `source` / `utm_source`, `campaign` / `utm_campaign`, `medium` / `utm_medium` sowie `track` / `courseLevel` / `f` (`GK` oder `LK`, Standardwert `GK`):

```
/start/abi26-he-mathe-k1?source=bluesky&campaign=abi26-mathe-k1&medium=social&track=GK
```

Beim Klick auf `Kostenlose SkillPilot-ID erstellen` läuft vierstufig ab:

1. Erzeugung einer neuen pseudonymen SkillPilot-ID (`POST /api/ui/learners`).
2. Setzen des Curriculums auf Hessen Sekundarstufe II Mathematik (`PUT /api/ui/learners/{id}/curriculum`).
3. Setzen des Einstiegs-Scopes auf `Abiturprüfung Mathematik` (`9ad83149-3cb7-5b87-a617-3eae3715a50c`) für `GK` bzw. `Abiturprüfung Mathematik (LK)` (`464a6024-a2f8-53b4-84e0-d7b9df22a0b1`) für `LK`.
4. Öffnen des Cockpits mit Kampagnenparametern (`l` auf `Gymnasium (DE)`, Bundeslandfilter `f = DE-HE`, Kursniveau `GK` oder `LK`).

## 14.2 Kampagnen-Events und QA-Prüfung (3 Test-Links)

Kampagnen-Events werden lokal in `localStorage` (`skillpilot_campaign_events`) gepuffert und an `POST /api/ui/events` übermittelt: `page_view`, `id_created`, `cockpit_opened`, `gpt_prompt_copied` und `gpt_start_clicked`.

Für den manuellen QA-Durchlauf sind drei Links zu prüfen:

```
1. GK:       /start/abi26-he-mathe-k1
               ?utm_source=bluesky&utm_campaign=abi26-he-mathe-k1
               &utm_medium=social&track=GK
2. LK:       /start/abi26-he-mathe-k1
               ?utm_source=bluesky&utm_campaign=abi26-he-mathe-k1
               &utm_medium=social&track=LK
3. Fallback: /start/abi26-he-mathe-k1
               ?source=bluesky&campaign=abi26-he-mathe-k1
               &medium=social&track=foo
```

Erwartetes Verhalten: Ungültiges `track=foo` fällt sichtbar auf `GK` zurück, die pseudonyme ID-Erstellung gelingt ohne Registrierung, das Cockpit öffnet mit gesetztem Scope sowie sichtbarem Coach-/GPT-Startblock, und der Feedback-Hinweis erscheint sowohl auf der Startseite als auch im Cockpit.