SkillPilot Concept Compendium · Volume III

Personalization, Learner Runtime, and Coach Architecture

Four-Level Personalization, Learning and Sync Workflows, Portable MCP-Host

Strategy, ChatClient/Backend Contract, and Plugin Lifecycle

Scope of this compendium

This volume unifies SkillPilot's ten normative runtime and coach specifications: the four levels of personalization, the end-to-end learning workflow and binary mastery contract, browser SRS synchronization and signed import/export, the long-term portable MCP-host strategy, the provider-neutral communication contract between ChatClient and backend, the coach product and adapter topology, strict separation between OAuth app authentication and 24-hour learning sessions, the behavioral integration Golden Journeys, and the Open AI plugin versioning and lifecycle architecture including the 19 October 2026 decision establishing the permanent `skillpilot-coach` identity.

As of 11 November 2026 · License: Apache-2.0 (technical architecture specification; see LICENSING.md).

Local candidates and specifications never substitute for real-host acceptance or portal publication.

Contents

1 Levels of Personalization **4**
1.1 Before Level 1: catalog navigation . . . 4
1.2 Level 1: Base Curriculum . . . 4
1.3 Level 2: Personal Curriculum . . . 4
1.4 Level 3: Learning Focus and Active Goal . . . 5
1.5 Level 4: Mastery and the orthogonal timeline axis . . . 5

2 Learning Workflow **6**
2.1 Data foundation: landscapes, structure, and filters . . . 6
2.2 Lifecycle steps 1–3: Base Curriculum, Personal Curriculum, and Focus . . . 6
2.3 Step 4: frontier calculation and evaluation modes . . . 6
2.4 Step 5: mastery, aggregation, and feedback . . . 7

3 Client-State Sync for Browser SRS Progress **8**
3.1 Endpoint and authentication boundary . . . 8
3.2 Request and response payloads . . . 8
3.3 Storage, privacy, and client behavior . . . 9

4 Import/Export Workflow **10**
4.1 Learner-state export, signature verification, and import . . . 10
4.2 Teacher-held local class files . . . 10

5 Portable Agent Runtime Architecture: Interchangeable MCP Hosts **12**
5.1 Part A — Durable architecture: three-tier separation and core invariants . . . 12
5.2 State boundaries, operating tracks, media, and billing . . . 13
5.3 Phased roadmap and mandatory host acceptance suite . . . 13
5.4 Part B — Dated market appendix (as of 19 August 2026) . . . 14

6 Communication Contract Between ChatClient and SkillPilot Backend **15**
6.1 Core boundary, responsibility matrix, and minimal turn contract . . . 15
6.2 Tools and inputs: semantically minimal, zero chat free text . . . 16
6.3 Outputs, opaque capabilities, and Verified Recall reference . . . 16
6.4 Choice, errors, privacy, anti-patterns, and 11-point gate . . . 17

7 SkillPilot Learning Coach: Product and Provider Architecture **19**
7.1 Product decision and multilingual plugin structure . . . 19
7.2 System topology and layer responsibilities . . . 19
7.3 Web-first onboarding, safe projection, and UI surface . . . 20
7.4 Grading boundary, multi-provider adapters, and 8 release gates . . . 20

8 OpenAI MCP App: OAuth and 24-Hour Learning-Session Binding **22**
8.1 Architectural Decision: Two Independent Credentials . . . 22
8.2 Mandatory Launch Sequence, Renewal, and Diagnostic TTL . . . 23
8.3 MCP Validation Order, Action Horizon, and Forbidden Fallbacks . . . 23
8.4 Persistence Schema, OAuth/mTLS Hardening, and Error Behavior . . . 24
8.5 Security Invariants and Automated Acceptance Criteria . . . 25

9 Behavioral Integration of the MCP Learning Coach **26**
9.1 Learner Experience Target and Non-Negotiable Invariants . . . 26
9.2 General 10-Step Decision Cycle, Scope vs. Focus, and Error Recovery . . . 26
9.3 Authoritative Rule Homes and Mandatory Behavioral Policies . . . 27

9.4 Golden Journeys (GJ-01 through GJ-11) 29
9.5 Test Pyramid, Quality Targets, Test Log, and Definition of Done 30

10 SkillPilot: Versioning and Lifecycle Architecture for the OpenAI Plugin 32
10.1 Decision of 19 October 2026: One Permanent Neutral Plugin 32
10.2 Independent Version Layers and SemVer Classification 33
10.3 Public Contract Adapters, Envelopes, and Dedicated MCP Apps UIs 34
10.4 Canonical contract-line/v1 Schema, CI Gates, Observability, and Line History 35

1 Levels of Personalization

This chapter specifies how SkillPilot narrows from the published curriculum catalog down to individual atomic mastery. Catalog navigation precedes the four personalization levels and is never persisted as learner scope.

These four levels remain distinct from Learning Units (*Lerneinheiten*, `LearningUnit`) — recursively composed didactic units with intended outcomes and derived completion whose placement belongs to the resolved educational context rather than forming a fifth personalization level.

1.1 Before Level 1: catalog navigation

Learners first choose a catalog area such as **school**, **university and higher education**, or **languages and continuing education** to locate an appropriate root curriculum. This step aids discovery only and stores nothing on the learner record.

1.2 Level 1: Base Curriculum

The learner selects one published root curriculum providing the complete goal universe from which subsequent levels narrow without copying or altering canonical goals (for example `Gymnasium (DE)`, a university degree catalog, or a CEFR language skill landscape).

Only root curricula explicitly listed in `curricula/curriculum_manifest.json` are selectable in the UI and eligible for Champion registration:

```
{
  "curricula": [
    { "id": "a0e13c56-c25f-4742-9272-3a1a603ee52e", "title": "Gymnasium (DE)" }
  ]
}
```

Every manifest ID must exist under `curricula/`, must never point to a module or contained sub-landscape, and is verified by CI against the computed root set.

1.3 Level 2: Personal Curriculum

The Personal Curriculum is the learner's committed, durable scope inside the Base Curriculum — defining which authored curriculum view applies rather than temporary study focus. For German school curricula, the semantic selection order is:

1. **Jurisdiction** (including an explicit canonical view where available);
2. **Duration model** (such as G8 or G9 where relevant);
3. **Stage** (lower secondary, upper secondary, or explicitly both);
4. **Selected subjects**;
5. **Course profile per selected subject** (where applicable).

Course profile is a per-subject attribute: Mathematics `LK` and Physics `GK` may coexist, `LK` never implies upper secondary, and missing stage data never silently defaults to cross-stage scope. Level 2 resolves the applicable composition view and bounds focus, frontier, progress, and completion; changing Level 2 revalidates focus and `activeGoal` without altering canonical goal IDs or erasing global mastery.

Single first-party editor contract: Level 2 is edited exclusively in SkillPilot's first-party Web GUI (on the start screen before launching a session and in the Cockpit afterward) over shared backend validation, normalization, and save rules. The OpenAI ChatClient neither presents nor mutates Level 2; if configuration is incomplete, it halts domain coaching and refers the learner back to the WebGUI.

1.4 Level 3: Learning Focus and Active Goal

Level 3 is intentionally lightweight and comprises two parts:

- **Level 3a — Focus:** one or more scope roots selecting the active learning corridor (such as Grade 7, Q1, or Analysis; structural program units are not themselves canonical goals).
- **Level 3b — Active atomic goal:** at most one concrete, assessable atomic goal serving as the immediate working target.

Focus widening follows the learner-facing ancestor path toward the root: the backend publishes valid broader ancestors that add unmastered `target` goals (nearest suitable ancestor first, preserving independent multi-root corridors in `goalIds`), and automatic widening is proposed only upon genuine completion of the current focus (never merely because the frontier is blocked), requiring learner confirmation.

Under the 4 November 2026 Assessment-Gated Completion clarification, `requires` alone never writes mastery to prerequisites; valid passed application assessments derive globally effective recognition of ordinary goals, so widening focus never reopens recognized competences.

1.5 Level 4: Mastery and the orthogonal timeline axis

Mastery is the learner's single durable outcome per stable atomic goal UUID — **mastered or not mastered** — remaining global across composition views and Level 2/3 changes without partial-mastery states. Effective mastery $M_{\text{eff}} = M \cup W$ combines stored explicit mastery M with assessment-gated recognition W (while assessment and memorization goals stay explicit; runtime migration remains pending, see 2.4). Cluster and scope percentages express the weighted share of effectively mastered atomic goals, never partial mastery of one goal.

Orthogonal curriculum timeline: A dated course or exam schedule is an execution-plan overlay answering **when** targets inside Level 2 should be addressed (see Unified Learning-Plan Status and Curriculum Time Axis and Pacing), not a fifth personalization level.

2 Learning Workflow

This chapter traces the operational runtime flow from static skill landscapes under `curricula/` to personalized navigation and binary mastery updates.

Inside the backend the workflow is provider-neutral. For ChatGPT (8), a registered OAuth client authenticates the integration while every explicit **Start Learning** (*Lernen starten*) click creates a fresh, absolutely expiring 24-hour `learningSessionId` injected into the launch prompt and passed unchanged to every domain MCP tool call; the permanent SkillPilot ID stays strictly backend-only. The Visible Session Flow remains an isolated rollback path, and the Claude lane uses the same state projection while awaiting full end-to-end client acceptance. Shared Chat Client/backend boundaries are governed by 6.

2.1 Data foundation: landscapes, structure, and filters

Each skill landscape under `curricula/` is a DAG with `contains` (hierarchy/composition) and `requires` (didactic prerequisites). During ongoing cluster-edge migration, runtime navigation derives **effective prerequisites** from both direct `requires` and `requires` inherited via `contains` ancestors, while Personal Curriculum filters narrow the visible learner view without preventing strict prerequisite checks across the structural graph.

2.2 Lifecycle steps 1–3: Base Curriculum, Personal Curriculum, and Focus

1. **Base Curriculum (Level 1):** Preceded by stateless catalog navigation, the learner selects one authority-defined root curriculum providing the complete goal universe.
2. **Committed Personal Curriculum (Level 2):** Configured solely in the first-party Web GUI (jurisdiction/canonical view → duration model → stage → subjects → per-subject course profile), producing the candidate space for focus, frontier, progress, and teacher course-plan selectors (which read Level 2 independently of mutable Level-3 focus).
3. **Focus and Active Goal (Level 3):** - **Level 3a (Focus roots):** Once Level 2 is complete, focus is never empty (defaulting deterministically to the first visible root if unset, while multi-root corridors remain supported for backend/coach plans). - **Level 3b (Active atomic goal):** At most one atomic goal active in the UI/coach loop. - **Plan-guided progression:** Confirmed teacher schedule activation atomically replaces included subject schedules, enables plan mode (suppressing generic Autopilot), and selects the first due frontier goal for an unfinished period quota; foreground subject switches park unfinished work without mutating mastery, and post-completion handoffs rank unfinished subject quotas deterministically without ever bypassing `requires` or inventing fallback goals (see Unified Learning-Plan Status and legacy Daily Learning Progress). - **Focus widening:** Offers backend-published broader ancestors along the visible path to root only after genuine completion of the current scope and learner confirmation.

2.3 Step 4: frontier calculation and evaluation modes

An atomic goal enters the **frontier** iff it is not yet mastered and all of its effective prerequisites are satisfied under their goal-type contracts. Under the 4 November 2026 Assessment-Gated Completion model (runtime adaptation pending), effective mastery $M_{\text{eff}} = M \cup W$ satisfies prerequisites without writing synthetic mastery rows, while ordinary mastery of B in $Q \to B \to A$ never recognizes A while local assessment Q remains open.

- **Optimistic mode:** Restricts prerequisite blocking to the active focus (computing transitive reachability prior to target masking so hidden intermediate nodes never sever paths), letting learners start inside their chosen scope.
- **Strict mode:** Keeps candidates inside the filtered scope while enforcing global prerequisite closure to surface foundational gaps outside the filter.

When a learner requests a non-frontier goal, the runtime traces missing effective prerequisites and redirects to those blockers.

2.4 Step 5: mastery, aggregation, and feedback

Every atomic goal has one authoritative persisted outcome on its stable UUID: **mastered or not mastered** (`0.0` or `1.0`; intermediate values are never a third pedagogical state).

- **Global persistence and aggregation:** Mastery stays global across view, Level-2, and Level-3 changes. Cluster/scope progress is the weighted share of unique effectively mastered atomic goals (60% means e.g. 6 of 10 goals mastered, never 60% of one goal).
- **Memorization (SRS) goals:** Derive their binary goal outcome from card state: ordinary flashcard practice in Cockpit or the dedicated MCP App component (`Not yet known` / `Known` with back/next navigation) updates SRS scheduling only, whereas **Verified Recall** is a separate unaided coach batch test. A memorization goal counts as mastered only once all required cards are verified and none is currently due (see Node Types). Orientation completion markers record finished onboarding dialogue without claiming subject competence.
- **Implementation status (verified 30 October 2026):** `LearnerService` accepts only `0.0` or `1.0` on new general mastery writes and `GoalCard.tsx` exposes binary controls, while `Mastery.java` retains `double` storage with the `>= 0.9` read compatibility threshold for historical rows. Browser SRS sync cannot fabricate verified recall, and signed imports preserve verified card state (see Completion Pilot). Whenever mastery changes, unlocked prerequisites, frontier options, and progress summaries recompute immediately.

3 Client-State Sync for Browser SRS Progress

This chapter specifies the UI-facing backend contract for synchronizing browser Spaced-Repetition-System (SRS) flashcard progress. **Verified Recall** updates internal card verification state through its own capability-bound atomic MCP batch workflow (6.3), never through this browser endpoint.

3.1 Endpoint and authentication boundary

```
GET /api/ui/learners/{skillpilotId}/client-state/{nodeId}
PUT /api/ui/learners/{skillpilotId}/client-state/{nodeId}
```

- **First-party browser lane:** Calls the UI endpoints above using the pseudonymous `skillpilotId` to persist per-node SRS scheduling periodically, on demand, and across export/import recovery while keeping payloads strictly PII-free.
- **MCP coach lane:** Authenticates the integration via OAuth and resolves the learner exclusively through the active 24-hour `learningSessionId` (8); `skillpilotId` is never an MCP tool argument or output. Legacy per-card Visible-Session routes belong solely to the isolated Visible Session Rollback Flow.

3.2 Request and response payloads

`PUT` sends `Content-Type: application/json` with a per-memorization-node (`nodeId`) card map:

```
{
  "updatedAt": "2026-02-02T19:30:00.000Z",
  "srsState": {
    "math_analysis_c01": {
      "id": "math_analysis_c01",
      "interval": 3,
      "easeFactor": 2.4,
      "repetitions": 2,
      "nextReview": 1706892870000,
      "lastReviewed": 1706806470000,
      "verifiedRecall": {
        "status": "passed",
        "attempts": 1,
        "failures": 0,
        "lastTestedAt": "2026-02-02T19:28:00.000Z",
        "passedAt": "2026-02-02T19:28:00.000Z"
      }
    }
  }
}
```

- `srsState` stores SRS scheduling fields (`interval`, `ef`/`easeFactor`, `repetition`/`repetitions`, `nextReview`, optional `lastReviewed`) alongside backward-compatible read-preserved `verifiedRecall` metadata (browser sync cannot forge new verified-recall passes; see 2.4).
- `GET` returns `200 OK` with `{ "updatedAt": "...", "srsState": { ... } }` (or `404 Not Found` when unpopulated, leaving browser storage active).
- `PUT` returns `200 OK` with `{ "status": "ok", "savedAt": "...", "storedKeys": 42 }`.

3.3 Storage, privacy, and client behavior

- **Persistence:** Stored per `(learner, nodeId)` JSON record (e.g. `learner_client_state`) using `updatedAt` ordering and zero personal identifiers (no names or emails).
- **Client triggers:** Auto-syncs every **20** reviewed cards and via explicit **Save** controls; if a network sync fails, the UI surfaces an error indicator while continuing local flashcard practice uninterrupted.

4 Import/Export Workflow

This chapter specifies the portability workflows for moving pseudonymous learner state between SkillPilot accounts or instances and for encrypting browser-local teacher class files at rest.

4.1 Learner-state export, signature verification, and import

Learner import/export provides **tamper-evident portability**:

- **Included:** selected curriculum, enabled modules/personalization, mastery state, planned goals, provenance (`copySources`), and client-side SRS state.
- **Excluded & guaranteed:** zero personal identifiers; exporting never mutates the source profile, and importing preserves the target profile's own `skillpilotId`.

1. **Export:** The backend composes an **HMAC-SHA256-signed** `serverExport` payload (computed over parsed values so JSON whitespace/key-order formatting remains safe while any value edit invalidates the signature), and the frontend wraps it with browser `clientData.srsState` inside a versioned `"2.0"` JSON envelope.
2. **Import:** The backend verifies the HMAC-SHA256 signature, applies curriculum settings, updates mastered/not-mastered outcomes on imported goals (leaving untouched goals intact; legacy numeric values remain storage compatibility rather than partial mastery, see 2.4), replaces planned goals, appends the source `skillpilotId` and timestamp to `copySources` provenance, and lets the browser re-key `srsState` to the target ID. Under the 4 November 2026 recognition contract, effective mastery is reconstructed from explicit outcomes and their fixed authored contracts without revoking valid recognition or enlarging old results via newer graph edges.

```
{
  "version": "2.0",
  "exportedAt": "2024-05-20T10:00:00.000Z",
  "serverExport": {
    "data": { "learner": {}, "mastery": {}, "plannedGoals": [], "copySources": [] },
    "signature": "<hmac-sha256>"
  },
  "clientData": {
    "srsState": { "srs_state_USERID_GOALID": {} }
  }
}
```

4.2 Teacher-held local class files

Local trainer classes use a **separate browser-encrypted portability lane** because they map student names/aliases to permanent SkillPilot IDs; they are never learner-state exports and are never signed or seen by the backend:

1. The teacher enters and confirms a passphrase of at least **15 characters** (with consistent Unicode normalization prior to key derivation).
2. One validated local `ClassSession` is serialized inside a versioned `skillpilot-trainer-class` payload.
3. **PBKDF2-SHA-256** (`600,000` iterations, fresh 16-byte salt) derives an **AES-256-GCM** key (fresh 12-byte IV, 128-bit authentication tag, fixed class-file AAD).

4. Only the encrypted envelope downloads as `skillpilot-class-YYYY-MM-DD.skillpilot` (deliberately omitting the class name from the filename).

Algorithm, iteration count, version, purpose, Base64 fields, ciphertext, and decrypted payload are strictly validated before entering browser storage: recognized encrypted envelopes never fall back to plaintext parsing, and wrong passphrases vs. tampered ciphertexts emit the identical non-technical error message. Legacy plaintext `skillpilot-class-*.json` files remain importable with an explicit prompt to re-export in encrypted form. Because decrypted existing-learner class files restore permanent SkillPilot IDs with full learner access (see Existing-Learner Teacher View), exported class files must be handled as bearer-secret containers; file encryption protects downloads at rest without encrypting unlocked browser local storage.

5 Portable Agent Runtime Architecture: Interchangeable MCP Hosts

Scope and Structure

Strategic architecture governing SkillPilot's long-term host and model-provider independence. Binding operational contracts for current coach adapters are specified in 6 through 10 and take precedence whenever details differ. **Part A** defines the durable normative architecture; **Part B** records the dated market assessment (as of 19 August 2026, complemented by subsequent repository decisions in 10).

5.1 Part A — Durable architecture: three-tier separation and core invariants

SkillPilot is structured long-term into three independently replaceable tiers:

1. **SkillPilot Core** as the durable, provider-neutral learning system,
2. an **interchangeable AI Chat Application / MCP Host**,
3. an **interchangeable Model Provider** (preferably addressed via an Open-Responses-conformant interface).

Two Co-Equal Architectural Rules

1. **SkillPilot Core must depend neither on a specific chat host nor on a specific LLM provider.**
2. **Per learner, SkillPilot Core persists exclusively structured, schema-validated learning and control state** — never chat messages, learner phrasing, free-text answers, student work, images, attachments, or model grading transcripts.

Learning conversations do not vanish by keeping Core narrow: they reside in the MCP Host and (when vision is used) at the chosen Model Provider. Because no single platform today satisfies every requirement simultaneously (hosted runtime, cross-device chats, direct user-to-provider billing, Open Responses, Remote MCP, Skills, MCP Apps UI, Agent Plugins, mobile UX, voice dictation, vision, and minor-student eligibility), SkillPilot deliberately supports two operating tracks.

Standards and their exact roles

Standard / Term	Normative role in SkillPilot
Open Responses (vs. proprietary *OpenAI Responses API* or emulated `/responses`)	Open specification between the Agent Runtime (MCP Host) and Model Provider/Gateway; conformance requires passing official acceptance tests plus SkillPilot's capability profile — never inferred from a `/responses` URL alone.
Model Context Protocol (MCP)	Standardizes connectivity between MCP Host/Client and **SkillPilot Core** as a Remote MCP Server over **Streamable HTTP**.
MCP Apps	Optional sandboxed HTML UI resources attached to MCP tools. Every SkillPilot tool supports plain-text output (universal fallback), `structuredContent`, and optional MCP Apps UI; no essential didactic step may depend solely on UI rendering.
Agent Plugins v1	Portable install artifact (`plugin.json`, `skills/.../SKILL.md`, `mcp.json`); MCP Apps UIs are served live by the MCP server, and OAuth client profiles stay host-managed outside the plugin archive.

Runtime and installation topology

```
End Devices (Mobile / Desktop Browser / App: Text, Dictation, Camera)
        | HTTPS
        v
AI Chat Application / MCP Host
  +-- Chat UI + Server-side Agent Runtime (Loop, Context, Attachments, MCP Client)
        |                                          |
        | Open Responses API / Gateway             | Remote MCP (Streamable HTTP + OAuth)
        v                                          v
Model Provider (Learner/Parent          SkillPilot Core
  account; vision images land HERE,       +-- MCP Server (Tools, Resources, MCP Apps UI)
  never inside SkillPilot Core)           +-- Didactic Services (Skill Graph, SRS, Exams)
                                         +-- Learning State DB (structured state only)
```

5.2 State boundaries, operating tracks, media, and billing

Processing domain	Stored data	Controller / owner
MCP Host	Chat user account, chat transcripts, messages, attachments, sessions, optional BYOK credentials	Host operator (in self-hosted pilot: enpasos)
Model Provider	Submitted prompts and vision images under provider terms	Model provider (direct contract with learner / parent)
SkillPilot Core	Structured learning and control state (learner link, curriculum/personalization, focus, `activeGoal`, binary mastery, SRS schedule, binary Verified Recall outcome)	enpasos

- **No cross-host chat portability:** Chat history is host-bound while learning state is Skill Pilot-bound (`learner`, `activeGoal`, `masteryValue`, `dueItems`, `lastRecallOutcome`; `nextRecommendedAction` is computed dynamically rather than persisted). Active coach mutations exclude free-text fields (retiring legacy `lastFeedback` exceptions in active contracts; see 6.2).
- **Visible identifiers:** A random, opaque `learningSessionId` is explicitly permitted inside model context, whereas internal learner, account, or database IDs remain strictly hidden.
- **Two durable operating tracks: Track A** uses provider-hosted MCP hosts (ChatGPT, Claude) with near-zero infra overhead and direct consumer subscriptions; **Track B** uses a portable open-source MCP host (pilot: single-node Docker Compose LibreChat; secondary lab: Archestra) with BYOK, direct provider billing, and explicit operator policies for Mongo DB backups, upload-volume retention, key rotation, and minor-protection access controls.
- **Dictation, ephemeral vision, and billing:** Voice input converts to text in browser/host; photographed student work travels from MCP Host to the chosen vision Model Provider and **never** to SkillPilot Core (Core receives only structured writes such as `setMastery` with `1.0`). Provider accounts and API credentials belong strictly to learners or parents/guardians — a central SkillPilot operator key reselling tokens is prohibited.

5.3 Phased roadmap and mandatory host acceptance suite

1. **Phase 1 — Stabilize interfaces:** Host-neutral Remote MCP surface (Streamable HTTP, OAuth 2.0, stable tool names, versioned schemas, idempotency, text fallback, `structuredContent`, optional MCP Apps) plus canonical Agent Plugin bundle with zero image/solution upload tools.

2. **Phase 2 & 3 — Self-hosted pilot & standards lab:** Closed LibreChat pilot (5–10 testers across phone/desktop, dictation, vision, BYOK) alongside practical evaluation of Archestra (Open Responses acceptance tests, MCP Apps).
3. **Phase 4 & 5 — Hosted channels & future managed host:** Claude (adult testers) and ChatGPT (directory release) sharing the same SkillPilot Core state; migration to a third-party managed portable host only once all 12 requirements (Open Responses, BYOK, MCP Apps, Agent Plugins, export/deletion, minor-eligible accounts) are proven.

Mandatory host and model acceptance gates: Cross-device session continuation without conflicting writes; 60 s German domain dictation triggering a tool call; phone-camera handwritten solution analysis with documented vision-provider retention; **MCP negative test** (proving zero image bytes, OCR text, solution prose, image URLs, or grading transcripts in MCP args, logs, traces, or DB); **credential negative test**; mobile OAuth refresh and opaque `learningSessionId` with zero internal ID leaks; write idempotency; account/chat deletion verification; complete text-only learning flow plus sandboxed MCP Apps check; direct user billing; official Open Responses test suite.

5.4 Part B — Dated market appendix (as of 19 August 2026)

Criterion	ChatGPT	Claude	LibreChat	Archestra	Target Managed
Runtime ops	OpenAI	Anthropic	self/3rd-party	self/3rd-party	3rd-party
Mobile MCP	verify in directory	yes (iOS/ Android)	mobile web	to verify	required
Voice & camera	dictation/ vision yes; live voice limited	yes	yes (vision direct to provider)	attachments doc.; mobile test open	required
Billing	ChatGPT sub	Claude sub	direct BYOK	personal keys	direct BYOK
Open Responses	internal proprietary	no	partial	`/responses` router to test	required
MCP + Apps	Remote MCP; dev mode web-only	Remote MCP + mobile Apps	Remote MCP + PKCE; Apps draft	Remote MCP + Apps	required + text fallback
Age policy	13+ (13–17 w/ consent)	Consumer 18+ (API separate)	operator-defined	operator-defined	minor-eligible
Role	strategic channel	adult benchmark	**immediate pilot**	**2nd candidate**	long-term goal

Open WebUI is not prioritized ahead of LibreChat or Archestra due to lacking native core MCP Apps support. Related normative contracts: 6, 7, 8, 9, 10, and OpenAI MCP Client Binding.

6 Communication Contract Between ChatClient and SkillPilot Backend

Status and Scope

Canonical, provider-neutral communication norm governing every SkillPilot learning coach (OpenAI MCP App, Claude Connector V1, Gemini Connector V1, future provider adapters, and all new coach tools). Security, session, didactic, and deployment specs refine this boundary without ever redefining it inconsistently. Passing tests on one provider lane (such as Gemini's controlled web check in the Gemini Integration Runbook) never implies acceptance of other hosts, accounts, or surfaces.

6.1 Core boundary, responsibility matrix, and minimal turn contract

One Sentence Defines the Boundary

The ChatClient understands and phrases; the backend knows, decides, validates, persists, and supplies the next technical step.

Language models excel at understanding natural language, spotting domain patterns, teaching dialogically, and comparing semantically equivalent solutions; the backend excels at identity, authoritative state, exact sets and ordering, permissions, transactions, concurrency, persistence, and reproducible state transitions.

Participant	Owns	Explicitly does not own
Learner	Learning intent, answers, natural phrasing, explicit consent	technical IDs, state versions, retry keys, tool call sequences
ChatClient / Model	Natural-language understanding, unambiguous mapping to fresh options, didactic dialogue, semantic solution comparison, phrasing	authoritative learning state, candidate sets, workflow progress, persistence receipts, authorization
Provider adapter	Compact use-case tool surface, safe projection, localized rendering, transport binding	separate domain DB, raw domain DTOs, prompt-only security guarantees
SkillPilot Backend	Identity, session, curriculum, personalization, focus, active goal, frontier, mastery, recall, exam unlock, IDs/capabilities, sets, order, state, concurrency, idempotency, persistence, continuation	open-ended natural-language parsing or semantic grading of visible learner work
App Widget	Direct deterministic interaction inside its bounded UI workflow	replacement for backend truth or general chat orchestration

Interaction language (`communicationLocale` on the latest authoritative full result) is likewise backend state — never guessed from tool names, host locale, or curriculum language.

Minimal turn contract

```
new learner message -> exactly one fresh full context -> decision:
  +-- speak / ask / bound UI action --> visible reply
  +-- at most one unambiguous mutation --> authoritative successor state
        +-- re-apply continuing unambiguous intent
        +-- speak / bound UI action --> visible reply
```

1. Load fresh context **exactly once** after a new learner message; intra-turn polling is forbidden.
2. The latest successful full state is the sole authority for language, state, options, and allowed actions.
3. Every successful mutation returns its complete successor state, replacing prior context for the rest of the turn (making extra context reloads redundant).
4. Narrow renderer or widget receipts are never full contexts.
5. On state conflict, reload context once and re-evaluate; repeated conflicts or hard errors halt structured execution transparently.

6.2 Tools and inputs: semantically minimal, zero chat free text

Tools model cohesive learner actions rather than 1:1 internal service calls: backend-derivable parameters stay out of model inputs, item loops become atomic batch operations, distinct permission/confirmation boundaries stay separate tools, and reads vs. writes carry truthful annotations.

Hard boundary: zero chat free text to Core The coach evaluates visible learner work and phrases feedback autonomously inside the provider chat — neither verbatim answers nor paraphrases or grading rationales are sent to the backend. Free-text fields for work, outcome, or recall feedback are forbidden in active coach mutations (unknown extra properties are rejected prior to domain invocation, replay hashing, and persistence):

- **Ordinary goal:** Structured completion decision → backend verifies active goal, eligibility, and state rules, returning goal, mastery, and successor state.
- **Verified Recall:** Exclusively the structured `passed` booleans for the authorized batch; learner answers and explanations stay in chat.
- **Assessment / Exam:** Authorized numeric score; rubric reasoning, partial-point commentary, and pass explanation stay in chat.

Didactic pacing: after finishing an exercise, the coach discusses it, offers questions or continuation, and waits; once sufficient evidence proves an ordinary goal or passed exam, the structured completion write fires immediately **before** claiming saved progress (whereas orientation and Verified Recall retain explicit learner consent prior to saving). Explanation, task, and goal image for a backend-selected successor start only after recognizable learner agreement to continue. Open questions never pre-populate suggested learner answers, and transport timeouts trigger clean recovery rather than invented "server unreachable" claims.

Model inputs carry only semantically mapped structured choices, semantic grading decisions (`passed`), or verbatim-copied fresh option payloads / opaque capabilities (plus `expectedStateVersion` and fresh `clientRequestId` on general V1 writes).

6.3 Outputs, opaque capabilities, and Verified Recall reference

- **Output channels:** `structuredContent` holds only model-needed facts; `content` at most a concise summary; `_meta` strictly widget-private data (never cross-turn memory).
- **Single imperative channel:** Ordinary full context uses `(requiredAction, instruction)`; a completed specialized workflow emits a single `continuation` (optionally carrying a server-filled `toolCall`). A parallel `presentationAction` or implicit UI binding on a domain write tool is forbidden.

- **Opaque capabilities:** Encapsulate session, workflow, and state authority (authenticated-encrypted whenever confidentiality matters), are forwarded verbatim by the model, and are validated server-side for completeness, staleness, and replay.

Verified Recall reference flow and terminal image handoff

```
start_skillpilot_verified_recall(learningSessionId)
  -> complete server-owned batch + batchCapability
get_skillpilot_verified_recall_answers(learningSessionId, batchCapability)
  -> complete ordered reference answers + gradingCapability
record_skillpilot_verified_recall_results(learningSessionId, gradingCapability,
assessments[{passed}])
  -> single atomic receipt + single continuation
```

When a terminal Recall write activates an ordinary atomic successor goal with an authorized visualization, the sole continuation is:

```
{
  "continuation": {
    "action": "renderGoalVisualizationThenTeachActiveGoal",
    "consentRequired": false,
    "toolCall": {
      "name": "render_skillpilot_goal_visualization",
      "arguments": {
        "goalId": "<server-filled successor goal>",
        "expectedStateVersion": 11
      }
    }
  }
}
```

The ChatClient copies tool name, server-filled `goalId`, and `expectedStateVersion` unchanged, adding only the already current unchanged `learningSessionId` under the global session gate — the Recall receipt deliberately does not mirror that session capability. The client invokes `continuation.toolCall` (`render_skillpilot_goal_visualization`) exactly once immediately and begins teaching the already active successor goal in the same turn without reconstructing image arguments from `goalVisualization` or `nextAllowedTools` and without reloading context (`reload context` is forbidden). If rendering fails or the host omits the widget, do not retry; full text instruction proceeds unchanged. Only the dedicated renderer binds image UI; the Recall mutation itself never binds image UI.

6.4 Choice, errors, privacy, anti-patterns, and 11-point gate

- **Explicit choice & errors:** Genuine choices publish a small fresh option set (copied verbatim when unambiguous; clarified only when genuinely ambiguous). Errors return stable codes with zero partial mutation and localized recovery instructions; session errors end exclusively with the first-party WebGUI start instruction, whereas widget rendering failures never block valid text coaching.
- **Privacy & anti-patterns:** Permanent SkillPilot IDs, OAuth tokens, client secrets, raw internal DTOs/paths, and unreleased solutions never enter model outputs. Forbidden anti-patterns include per-card/per-goal loops, optional parameters duplicating backend state, competing imperative fields, redundant post-mutation context reads, prompt-only safety rules, model-synthesized IDs/URLs, unreceipted save claims, exact-string backend NLP, and any chat free text in tool inputs or persistence.

- **Single rule home & 11-point acceptance gate:** Every rule has one authoritative home (this contract, 9, 7, 8, Unified Learning-Plan Status, `SKILL.md`, tool schema, backend guard, and `docs/deploy/`). Every interface change must pass all **11 gate questions**: (1) one cohesive user action per tool, (2) zero backend-derivable inputs, (3) zero model loops, (4) single authoritative continuation, (5) backend rejection of incomplete/duplicate/stale/unauthorized calls, (6) exact retry vs. new-operation distinction, (7) receipt-backed state claims, (8) language/semantic judgment in ChatClient, (9) valid/invalid/adversarial contract tests, (10) real provider trace with zero extra polls, (11) complete exclusion of chat prose from active inputs, echoes, and storage.

7 SkillPilot Learning Coach: Product and Provider Architecture

Status and Contract Map

Durable product and system topology of the multilingual SkillPilot Coach. Turn boundaries and input/output rules live in 6, OAuth and 24h session binding in 8, visible coaching behavior in 9, versioning including the 19 October 2026 decision establishing the permanent neutral `skillpilot-coach` identity (`https://mcp-openai.skillpilot.com/mcp`) in 10, and operational runbooks under `docs/deploy/`.

7.1 Product decision and multilingual plugin structure

SkillPilot delivers its coach as a provider-hosted conversation backed by authenticated SkillPilot tools:

- Model inference, open dialogue, and optional widget UIs run on the provider using the learner's own consumer account; SkillPilot invokes no paid model API on this path and never resells model tokens.
- Curriculum, personalization, focus, frontier, active goal, mastery, recall, assessments, and persistence remain entirely inside the SkillPilot backend.

The multilingual OpenAI plugin (`skillpilot-coach` under the neutral cutover decision in 10.1; historically `skillpilot-coach-v1` in the preserved `1.1.2` Git beta) bundles:

- one neutral-English control-plane skill,
- one directly bound MCP server on a stable public origin (`https://mcp-openai.skillpilot.com/mcp`),
- one language-neutral tool catalog,
- two independently bound, content-addressed MCP Apps UI resources (goal visualization and flashcard practice),
- separate real-host acceptance suites per released interaction language and client surface (German, English, and future languages share the same plugin via the session-bound `communicationLocale`).

7.2 System topology and layer responsibilities

```
SkillPilot WebGUI (configure Level 2, click "Start Learning")
  |  fresh learningSessionId + prepared launch prompt
  v
ChatGPT: SkillPilot Coach
  |  neutral skill, neutral tool catalog, optional MCP Apps UIs
  |  HTTPS + OAuth Bearer + learningSessionId
  v
https://mcp-openai.skillpilot.com/mcp
  v
OpenAI V1 Provider Adapter inside Spring Boot (/internal/openai/v1/mcp)
  |  AuthN/AuthZ, rate limits, JSON schemas, safe state projection
  v
CoachToolFacade + CoachStateProjection
  v
LearnerService + Curriculum + Database
```

Provider adapters, domain services, and persistence run inside the single `skillpilot-server` deployment with no separate gateway process.

Layer	Responsibility
First-Party WebGUI	Permanent ID, provider notice, Level-2 editor, explicit session launch, and Cockpit fallback
Provider Host	LLM, chat UI, skill activation, tool selection, and optional sandboxed widget rendering
Plugin Package	Installable bundle of manifest, skill tree, MCP endpoint config, and release metadata
Coach Skill	Repeatable pedagogical dialogue behavior without domain-state authority
Provider Adapter	OAuth/scopes, tool catalog, schemas, rate limits, safe projection, and provider UI metadata
`CoachToolFacade` & `CoachStateProjection`	Compact provider-neutral domain use cases and explicit allowlist of model-visible state
Domain Runtime	Curriculum, state machine, valid transitions, transactions, idempotency, and persistence

7.3 Web-first onboarding, safe projection, and UI surface

- **Web-first Level 2:** Permanent SkillPilot ID, base curriculum/view, jurisdiction, duration model, stage, subjects, and course profiles are configured exclusively in the first-party Web GUI; the V1 coach contract neither asks for nor mutates them in chat, referring incomplete profiles back to the WebGUI while keeping Level-3 focus and `activeGoal` adjustable via fresh backend options.
- **OAuth vs. 24h learning session:** OAuth authenticates the integration; every confirmed WebGUI launch independently mints an absolutely expiring 24-hour `learningSessionId` (never extended by OAuth token refresh; see 8).
- **Safe state projection:** Providers never receive raw `UnifiedLearnerStateResponse` or internal domain entities: permanent IDs and copy-provenance are stripped, tasks are separated from solutions/rubrics (unlocked only inside the protected active evaluation use case), URLs/math are normalized, model facts go into `structuredContent`, and widget-private data stays inside `_meta`.
- **Two dedicated MCP Apps resources:** `render_skillpilot_goal_visualization` binds the active content-addressed image resource; `start_skillpilot_memory_practice` binds the interactive flashcard resource (`release/line.json` is the single source of active hashes while previously served image URIs remain passively readable byte-for-byte). App-only card review and all other tools stay UI-unbound, and the text-only learning path remains self-sufficient.

7.4 Grading boundary, multi-provider adapters, and 8 release gates

The provider model compares visible learner work semantically against backend-released reference solutions/rubrics, while the backend alone governs release eligibility, active goal/attempt identity, batch completeness, stale/replay rejection, and state updates. Additional providers (such as Claude or Gemini) receive their own auth/transport adapters, reviewed tool catalogs, and dedicated end-to-end acceptance runs atop the shared `CoachToolFacade` and `CoachStateProjection`.

Before public release, **all 8 gates** must pass together: (1) contract, schema, security, and domain suites, (2) byte-exact source/draft parity, (3) OAuth, session, expiry, and recovery checks, (4)

real Golden Journeys on the live provider model, (5) desktop, mobile-web, and target native surfaces, (6) every released interaction language, (7) privacy, minor-protection, support, and submission checks, and (8) telemetry, rollback readiness, and intact Cockpit usability.

8 OpenAI MCP App: OAuth and 24-Hour Learning-Session Binding

Scope and current baseline

Authoritative identity and session-binding specification for the multilingual OpenAI MCP coach (**SkillPilot Coach** / `skillpilot-coach` on `https://mcp-openai.skillpilot.com/mcp` per the 19 October 2026 decision in 10.1; historically `skillpilot-coach-v1` on `https://mcp-coach-v1.skillpilot.com/mcp` in the preserved `1.1.2` Git beta). For plugin identity, contract major, and lifecycle rules, see 10.

8.1 Architectural Decision: Two Independent Credentials

SkillPilot separates integration authentication from learner addressing through two independent credentials:

1. **OAuth authenticates and authorizes the MCP integration:** The hosted directory app uses a single pre-registered confidential OAuth client (`client_id` + high-entropy `client_secret` via `client_secret_basic` at the token endpoint, Authorization Code + PKCE `S256`). For the authorized native desktop route, a separate, strictly pinned public CIMD profile (`https://chatgpt.com/oauth/codex/7yTcOAOqrumX/client.json`, loopback callback `http://127.0.0.1/callback/7yTcOAOqrumX` on a dynamic port with PKCE `S256`) exists alongside it and never serves as a fallback from failed confidential authentication.
2. **A temporary 24h learning session selects the learner and interaction locale:** Only an explicitly confirmed **Start learning** (**Lernen starten**) action in the first-party WebGUI mints a fresh opaque `learningSessionId` (`PT24H`) and opens a new chat. The mapping to the permanent SkillPilot ID and `communicationLocale` remains exclusively inside the backend.

Every domain MCP tool call requires both credentials simultaneously:

```
valid OAuth access token (exact resource + read/write scope)
AND
valid learningSessionId (active + remaining lifetime >= PT1H)
```

OAuth alone never creates a learning session or selects a learner; a `learningSessionId` alone never authorizes an MCP call.

Binding	Transport	Server-side meaning
ChatGPT/App -> SkillPilot	OAuth code flow (`client_secret_basic` or pinned desktop CIMD + PKCE `S256`), then `Authorization: Bearer`	Only the approved client profile can obtain and present an access token audience-restricted to the active MCP resource.
Chat -> Session -> Learner	`learningSessionId` as a required input on every domain MCP tool	This active first-party session addresses exactly one authoritative learner state and `communicationLocale`.

The permanent SkillPilot ID never leaves the backend in the MCP flow and never appears in chat messages, tool arguments, or tool responses.

8.2 Mandatory Launch Sequence, Renewal, and Diagnostic TTL

Every click on **Start learning** in the first-party WebGUI is an atomic launch operation:

1. The WebGUI holds the selected SkillPilot ID, chosen `communicationLocale`, and configured Level 2/Level 3 context.
2. It sends a single launch request to the SkillPilot backend.
3. The backend atomically applies the typed launch context to the authoritative learner state.
4. At that exact moment, it generates a fresh cryptographically random `learningSessionId`.
5. It stores only the HMAC hash together with internal `learner_id`, `communication_locale`, `started_at`, and absolute `expires_at`.
6. It returns a prepared bootstrap prompt and launch URL; the WebGUI opens the new chat pre-populated with that message:

```
Use SkillPilot Coach and continue.
learningSessionId: sps_<random opaque value>
```

- **Every launch is independent:** Every launch request mints a distinct `learningSessionId` — even when the same learner restarts immediately, when an older session is still valid, when another learner shares the same ChatGPT account, or when OAuth is already connected. Concurrent sessions remain independently valid; usage never extends `expires_at`.
- **Renewal exclusively via first-party WebGUI:** Without a current bootstrap message, the model calls zero SkillPilot tools and instead points succinctly in the user's language to `https://skillpilot.com/` and the WebGUI launch flow. When a tool returns `SESSION_REQUIRED`, `SESSION_RENEWAL_REQUIRED`, or `SESSION_VERSION_UNAVAILABLE`, the coach outputs `instruction` verbatim (or the exact localized entry from `instructions`), appending `startUrl` only if not already included — with no domain teaching, no reuse of the old session, and no OAuth reconnection.
- **Gated first-party diagnostic TTL:** Exclusively `POST /api/ui/learners/{skillpilotId}/openai/v1/launch` may accept optional `diagnosticSessionTtlSeconds` when `SKILLPILOT_OPENAI_COACH_V1_DIAGNOSTIC_SESSION_TTL_ENABLED=true` and the integer lies within `3601..86400` seconds (never exceeding `PT24H`). Shortened lifetimes (e.g., `3660` s to observe the `< PT1H` cutoff after 1 minute, or `5400` s for a 90-minute soak) apply solely to that single session; omitting the field immediately restores `PT24H`.

8.3 MCP Validation Order, Action Horizon, and Forbidden Fallbacks

The model copies `learningSessionId` unchanged into every domain tool call as a plain required string field (keeping regex and length rules server-side so the LLM never fabricates tokens), while the connector sends `Authorization: Bearer <token>` outside model context. On every call, the backend validates in this exact **8-step order**:

1. valid OAuth access token;
2. exact Resource/Audience (`https://mcp-openai.skillpilot.com/mcp`) and required read or write scope;
3. present, syntactically valid `learningSessionId`;
4. HMAC lookup in `openai_de_learning_session`;
5. non-expired, non-revoked session status;
6. authoritative learner resolution and availability of the pinned contract, workflow, and curriculum revisions;

7. remaining lifetime `expiresAt >= now + PT1H` for every read, write, or replay response (exact `PT1H` is valid; below `PT1H` fails with `SESSION_RENEWAL_REQUIRED`);
8. on writes, check for an exact committed idempotent replay (identical tool name, canonically normalized arguments, identical `clientRequestId`, available pinned versions, and `completed StateVersion == current learner revision`; if canonical learner state has since advanced, fail with `STATE_VERSION_CONFLICT`).

Explicitly forbidden fallbacks: A missing or invalid `learningSessionId` must never fall back to an OAuth subject, the "most recent" or any other session of the learner, a pending launch, a SkillPilot ID typed in chat, or a session implicitly created during an OAuth callback. OAuth callbacks, token refreshes, and MCP reconnects never create, extend, replace, or reactivate a learning session. Shared household or classroom ChatGPT accounts therefore remain strictly isolated per launch.

8.4 Persistence Schema, OAuth/mTLS Hardening, and Error Behavior

Table `openai_de_learning_session` is the sole persistence boundary for short-lived session bindings (retiring earlier OAuth-subject primary keys):

Column	Meaning
`token_hash`	HMAC hash of the issued `learningSessionId` (plaintext is never stored)
`learner_id`	Server-internal foreign key referencing the learner record
`communication_locale`	Launch-pinned locale governing backend payloads and visible coach replies
`started_at`	Timestamp of the confirmed **Start learning** action
`expires_at`	Absolute expiration (`started_at + PT24H`, or `+ diagnosticSessionTtlSeconds` in gated tests)

OAuth client secrets reside strictly in protected provider and server configuration (never in repositories, browser bundles, prompts, tool arguments, responses, or logs). Open Dynamic Client Registration (DCR) and unpinned CIMD documents are disabled; exact callback allowlists, PKCE `S256`, Resource/Audience, and scopes are enforced. In addition, the dedicated MCP v Host validates the OpenAI-managed client certificate (`observe` mode during initial integration passes missing certificates to OAuth while rejecting invalid ones; `enforce` mode requires the verified OpenAI chain, `clientAuth` EKU, and exact connector SAN; see OpenAI MCP Client Binding). mTLS never replaces OAuth client authentication or `learningSessionId`.

Failure condition	Mandatory behavior
OAuth missing / invalid	Standard host OAuth re-authorization; creates zero learning sessions
`learningSessionId` missing / unknown / tampered / expired / revoked	`SESSION_REQUIRED`; coach outputs verbatim `instruction` (plus `startUrl` if absent) with zero domain teaching; fresh WebGUI launch + new chat
Remaining lifetime < `PT1H`	`SESSION_RENEWAL_REQUIRED` (exact `PT1H` remains valid); fresh Web GUI launch + new chat
Pinned workflow / curriculum revision unavailable	`SESSION_VERSION_UNAVAILABLE`; fail-closed, fresh WebGUI launch + new chat
Write scope missing	Reject before any partial domain mutation
Launch context cannot apply atomically / invalid diagnostic TTL	Reject launch request; issue zero `learningSessionId`

8.5 Security Invariants and Automated Acceptance Criteria

The architecture enforces **17 security invariants**: (1) server-only permanent SkillPilot ID, (2) random opaque session IDs stored solely as HMAC hashes, (3) absolute `PT24H` lifetime except gated `3601..86400` diagnostic launches, (4) zero sliding expiration on use, (5) `PT1H` action horizon for new operations, (6) identical `PT1H` and learner-revision guards for idempotent write replays, (7) fresh independent session per launch, (8) no learner selection via OAuth alone, (9) no MCP authorization via session alone, (10) mandatory `learningSessionId` on all domain tools, (11) zero OAuth-subject lookup or fallback, (12) unchanged skill-graph, frontier, and mastery semantics, (13) strict client-profile separation (`client_secret_basic` vs. pinned desktop CIMD with zero fallback), (14) secret-free code/UI/prompts/logs, (15) launch-fixed `communicationLocale`, (16) mandatory fresh `get_skillpilot_context` at the start of every learner turn with direct in-turn use of mutation successor states, and (17) first-party WebGUI ownership of permanent IDs, provider notices, and Level 2 configuration.

Automated tests must verify **all 20 acceptance criteria**:

1. One UI launch click triggers exactly one backend launch request and opens ChatGPT.
2. Every launch mints a distinct `learningSessionId`, including repeated launches for the same learner.
3. Concurrent launches for different learners operate without interference over one OAuth connection.
4. The bootstrap message carries exactly one `learningSessionId` and zero SkillPilot IDs.
5. Every domain MCP tool schema requires `learningSessionId`.
6. A valid OAuth token without `learningSessionId` is rejected.
7. A valid `learningSessionId` without a valid OAuth token is rejected.
8. Unknown, tampered, revoked, or expired sessions fail closed with zero fallback.
9. Successful tool calls resolve the learner solely from the explicit `learningSessionId`.
10. Repeated tool usage leaves `expires_at` unchanged.
11. Domain read and write scopes are enforced per tool.
12. Existing goal, frontier, and mastery test suites remain green.
13. German and English share the same V1 tool contract governed by `communicationLocale`.
14. Token requests with missing/wrong secrets, foreign `client_id`, unapproved callbacks, or wrong resources are rejected.
15. Authorization-server metadata advertises the active production profile without open DCR or unpinned public fallbacks.
16. The diagnostic TTL field accepts `3601` and `86400` when gated on, and rejects `3600`, `86401`, values above `PT24H`, and any value when gated off.
17. Diagnostic launches (`3660` / `5400`) affect only their own session; the next normal launch defaults back to `PT24H`.
18. Every learner turn starts with a successful context read (session errors block all domain prose), and mutations return authoritative successor state used directly in the same turn without redundant reads.
19. The public V1 tool catalog exposes zero sessionless bootstrap, capability, curriculum, or personalization tools.
20. Write replays below `PT1H`, with unavailable pinned workflow/curriculum versions, or after canonical learner-revision advancement are rejected even when tool name, canonical arguments, and `clientRequestId` match.

9 Behavioral Integration of the MCP Learning Coach

Mission and scope

Living normative guide governing the **cohesive learner-visible behavior** and end-to-end acceptance of the multilingual MCP learning coach: uniting backend, skill graph, learner state, MCP contract, and provider model into one natural, dependable coach. Valid OAuth, clean MCP schemas, and available tools alone do not prove end-user reliability — only the complete user journey, actual tool trace, canonical backend state, and visible reply together count as evidence. Individual defects must yield general improvements rather than curriculum-specific workarounds. Companion contracts: 6, 8, 7, OpenAI MCP Client Binding, and Deployment Runbook.

9.1 Learner Experience Target and Non-Negotiable Invariants

Learners must never need to understand OAuth, MCP, tool arguments, or session tokens: (1) click **Start learning** (**Lernen starten**) in the first-party WebGUI, (2) land in a fresh chat with a pre-filled bootstrap message, (3) state learning goals in natural language, (4) continue directly inside the WebGUI-configured Level 2 context (the chat never asks for or mutates Level 2 settings such as curriculum, state/jurisdiction, duration/grade model, stage, subjects, or course profiles), (5) receive clarifying questions only for genuine pedagogical or Level 3 ambiguity, (6) work on the right atomic goal in the right mode, and (7) resume reliably after interruptions.

Invariants preserved across every provider, model, or runtime change:

- **Backend and skill graph:** The most recently loaded backend state is the sole source of truth; the coach never invents curricula, goals, options, progress figures, or successful mutations. Mastery attaches to the globally unique goal UUID across all composition views. Directed `requires` edges govern prerequisites (mastering a dependent goal never retroactively proves mastery of its still-open prerequisites) and `contains` governs composition; orientation/motivation goals are prerequisites only where the skill graph explicitly declares them. Frontier and recommendations are computed strictly after scope and focus are confirmed.
- **App binding and learning session:** OAuth authenticates the integration; every confirmed **Start learning** click independently mints a fresh, absolutely expiring 24-hour `learningSessionId` (without revoking older sessions or deriving identity from OAuth or prior chats). Every domain call and exact idempotent write replay requires at least `PT1H` remaining lifetime (exact `PT1H` is valid), available pinned workflow/curriculum versions, and — for replays — an unchanged canonical learner revision.
- **Learner communication:** Zero tool, API, JSON, or field names in normal replies; zero permanent SkillPilot IDs, OAuth tokens, or client secrets in chat; zero success claims before a confirmed backend receipt; zero hints or follow-up questions during active exams.

9.2 General 10-Step Decision Cycle, Scope vs. Focus, and Error Recovery

On entry, resumption, uncertainty, or potential context compaction, the coach executes a state-driven **10-step cycle**:

1. **Adopt learning session:** exclusively from the current SkillPilot bootstrap message.
2. **Load fresh state:** call context at the start of every learner turn inside the current assistant turn (never reconstruct state from chat memory; after a successful mutation, its complete successor state is authoritative for the rest of that same turn without redundant re-reading).

3. **Establish confirmed context:** distinguish confirmed backend state, fresh backend options, and unverified user intent.
4. **Evaluate permissible intent:** allow only domain work and explicit Level 3 focus/goal updates — never Level 2 setup.
5. **Map unambiguous next option semantically:** match solely against options in the freshest state by meaning rather than surface wording or list index.
6. **Execute at most one permitted mutation:** never pre-empt downstream choices or fabricate IDs.
7. **Adopt fresh successor state:** directly from the mutation response in the same assistant turn.
8. **Re-apply intent to the successor state:** advance through unambiguous Level 3/domain steps without unnecessary intermediate prompts.
9. **Ask only genuine residual ambiguity:** bundle related open questions concisely after stating confirmed context.
10. **Start pedagogical work only afterward:** frontier selection, coaching, recall, or exam begins strictly on confirmed scope and focus.

Scope vs. focus: The coach separates entry context (curriculum/program package), personalization (jurisdiction/applicability), learning scope (applicable target subset), focus (currently active subtree supplying frontier and goal options), and the single active atomic goal. Saying "Mathematics LK" sets subject and per-subject course profile in the WebGUI (e.g., Mathematics LK alongside Physics GK), without implying Sekundarstufe II vs. I, G8/G9, or phase; the coach must never leave focus on the full subject root and leak unconfirmed stages. For focus broadening, the backend publishes only the learner-facing ancestor path that adds open `target` goals (nearest ancestor first); automatic broadening is offered only after genuinely completing the current focus (an empty frontier alone is insufficient) and applied only after learner consent.

Bounded error recovery: On state conflict, reload once and re-evaluate intent; on `SESSION_REQUIRED`, `SESSION_RENEWAL_REQUIRED`, or `SESSION_VERSION_UNAVAILABLE`, emit verbatim `instruction` (or localized `instructions` entry) plus `startUrl` if absent, with zero domain teaching and zero OAuth reconnection; on auth, schema, storage, or repeated conflict failures, halt transparently without inventing fallback curricula.

9.3 Authoritative Rule Homes and Mandatory Behavioral Policies

Every rule has a single primary home; deterministic counting, ordering, completeness, and idempotency belong in atomic backend operations rather than longer prompts (frozen Custom GPT and Visible-Session bundles remain historical rollback baselines only):

Rule category	Authoritative home
Language vs. technical orchestration	Communication contract (6)
Learner-visible pedagogical behavior	This chapter (9)
Repeatable model behavior	Shipped `SKILL.md` and `references/coaching-policy.md` (`ai/openai plugin/skillpilot-coach/`)
Cross-tool protocol invariants	Concise `SERVER_INSTRUCTIONS` in `OpenAiDeV1McpContractAdapter`
Tool selection & preconditions	Tool name, description, and JSON schema in the V1 adapter
State-dependent step options	Single fresh payload from `OpenAiDeCoachContextProjector` / workflow adapter
Domain & security guarantees	Domain service, guard, transaction, and automated test
Real end-to-end behavior	Golden Journey, tool trace, backend state, and visible reply

Policy ID	Normative rule
`COACH-STATE-001`	Freshly loaded backend state is the sole state authority.
`COACH-SESSION-001`	The active learning session is passed unchanged to every domain call and never derived from OAuth or older chats.
`COACH-SESSION-002`	Operations and stored write replays require `>= PT1H` remaining lifetime (exact `PT1H` valid), available pinned workflow/curriculum versions, and unchanged canonical learner revision on replay without re-mutating.
`COACH-SESSION-003`	Session errors produce only verbatim server instruction plus `startUrl` if absent — zero domain replies or OAuth reconnects; continuation uses Web GUI launch and a fresh chat.
`COACH-BOOTSTRAP-001`	Without a current prepared session, the coach names only `https://skillpilot.com/` and the WebGUI launch flow and calls zero SkillPilot tools.
`COACH-INTENT-001`	Multi-part natural user intents persist across turns regardless of phrasing or order.
`COACH-CONTEXT-001`	Before asking open questions, the coach briefly states already confirmed domain context.
`COACH-SCOPE-001`	Level 2 scope and course profiles are resolved exclusively in the WebGUI and used only from fresh backend state.
`COACH-FOCUS-001`	Goal options originate strictly from the confirmed active focus without leaking earlier stages.
`COACH-MUTATION-001`	At most one currently allowed option is mutated per fresh state; its complete successor state governs the remainder of that assistant turn.
`COACH-QUESTION-001`	Only genuine residual ambiguities are asked, bundling related open items together.
`COACH-GOAL-001`	Teaching always addresses a single confirmed atomic goal.
`COACH-MASTERY-001`	Mastery follows the globally unique goal UUID and is recorded only on sufficient evidence.
`COACH-RECALL-001`	The backend owns card IDs, batch count, order, completeness, status, idempotency, and continuation; the model presents the full batch, awaits all answers, loads expected answers once, compares semantically, and commits all assessments atomically once without per-card loops.
`COACH-EXAM-001`	Exam mode presents the verbatim task, gives zero hints or follow-ups, and grades fairly against rubric criteria including valid alternative methods.

Policy ID	Normative rule
COACH-RESOURCE-001	Domain resources come solely from fresh state: freshly authorized goal visualizations render immediately once via their bound renderer, normal flashcard practice launches its bound component, and Cockpit fallback appears only in documented cases.
COACH-ERROR-001	Error handling is bounded, truthful, and never fabricates substitute learning flows.
COACH-PRIVACY-001	Technical identities, secrets, and internal IDs never appear in learner-visible replies.

10 abstract error classes guide general root-cause fixes instead of one-off patches: (1) Session/ Handoff, (2) State Projection, (3) Intent Understanding, (4) Scope Resolution, (5) Focus Selection, (6) Tool Intelligibility, (7) Orchestration, (8) Context Loss, (9) Didactics/Mode, and (10) Learner Communication.

9.4 Golden Journeys (GJ-01 through GJ-11)

- GJ-01 – **Fresh launch:** Clicking **Start learning** in the WebGUI mints a fresh 24h session, opens a new chat with a prepared prompt, and fetches context in the first assistant turn before any domain reply (no second click, manual copy-paste, or generic syllabus prose).
- GJ-02 – **Level 2 change requested in chat:** Inputs such as "Hesse, Math LK" trigger zero chat-side curriculum/personalization mutations or Level 2 setup prompts; the coach points succinctly to the server-supplied WebGUI route and resumes cleanly after a fresh launch.
- GJ-02a – **Multiple WebGUI-configured course profiles:** When e.g. Mathematics LK and Physics GK are stored, the coach reads both per-subject profiles unchanged from fresh context without assuming one global profile or inferring stage from LK/GK.
- GJ-03 – **Orientation and focus:** When a new active goal is confirmed, the first substantive reply opens with its exact localized `activeGoal.title` (never substituting the description); goal options come strictly from the focused subtree without stage or root leakage.
- GJ-03a – **Active motivation dialogue:** Turn 1 states the exact goal title, 2–4 age-appropriate possibilities, honest positive perspectives, and a low-threshold personal interest question with no right/wrong answer. Turn 2 builds on the chosen interest (e.g., "smartphones and AI") with 1–2 concrete things to understand or build plus a personal follow-up while keeping the orientation goal active. Completion follows the learner's response (or explicit request to move on): positive summary, invitation for questions/completion, and write only after consent (technical completion marker, never a domain-competency claim; zero knowledge quizzes or Feynman teach-backs).
- GJ-04 – **Standard coaching cycle:** One active atomic goal, prior-knowledge check, small hints, independent work, transfer, and fair reconstruction of alternative methods. Before replying, the coach judges from independent evidence whether only the task or the entire goal is complete (one solved task alone does not prove full goal mastery; when goal evidence suffices, mastery is persisted immediately before announcing success, even if the learner pauses; learner consent gates the next task/image rather than mastery persistence). Resumption continues the exact confirmed pedagogical step.
- GJ-05 – **Verified Recall:** Prepared batch (tested with 8, 5, and 10 cards): exactly one `start_skillpilot_verified_recall` call with no model-chosen `goalId` or `batchSize`; all returned questions shown in exact server order; all answers awaited; exactly one `get_skillpilot_verified_recall_answers` call; semantic grading accepting equivalent phrasing, batch feedback, and learner consent before committing; then exactly one atomic `record_skillpilot_verified_recall_results` call submitting one boolean per card — with zero per-card loops,

zero manual mastery writes, and zero extra context reloads in that turn. When the learner agrees to continue, the single backend `continuation` is executed unchanged (on pause, no new content starts). When the terminal write activates a successor goal with an eligible visualization, `continuation.action` is `renderGoalVisualizationThenTeachActiveGoal`: after learner agreement to continue, the coach executes the fully server-filled `continuation.toolCall` for `render_skillpilot_goal_visualization` with `goalId` and `expectedStateVersion` unchanged exactly once, adds only the already current unchanged `learningSessionId` per the session gate (never mirrored in the receipt), and then begins teaching the active goal — without deriving image arguments from successor context, reloading context, retrying on renderer/host failure, introducing a sibling `presentationAction`, or binding the Recall write itself to the image UI.

- `GJ-06` – **Exam mode:** Verbatim task, zero hints/questions during work, pass threshold and rubric disclosed only after full submission, required drawings produced as actual drawings, criterion-by-criterion grading honoring valid alternative solutions. Passing attempts record mastery immediately with the released evaluation capability and score (even if pausing) before announcing success; failed attempts write nothing and allow unlimited retries. Missing required authoritative figures pause the exam without invented visual details.
- `GJ-07` – **Long dialogue and rehydration:** Across 10–20 turns, tangents, reloads, or context pressure, the active session reloads the same backend state without mutating stale options.
- `GJ-08` – **Action horizon, expiry, and renewal:** Below `PT1H` (exact `PT1H` remains valid), after expiry/revocation, or on unavailable pinned revisions, the coach outputs only the verbatim server instruction (plus `startUrl` if absent) with zero domain replies, ID prompts, or OAuth reconnects; WebGUI launch starts a fresh session and chat.
- `GJ-09` – **Concurrent learners and chats:** Distinct UI launches and SkillPilot IDs stay strictly isolated.
- `GJ-10` – **Cross-curriculum generality:** The identical decision cycle works across school forms, states, and subjects with zero curriculum-specific code branches.
- `GJ-11` – **Bound UI actions and Cockpit boundary:** Every newly authorized `(goalId, stateVersion)` pair immediately invokes the goal-visualization renderer once; normal card practice launches its bound card component; Cockpit fallback appears solely on actual component failure, missing capability, explicit learner request, or server instruction.

9.5 Test Pyramid, Quality Targets, Test Log, and Definition of Done

The **5-layer test pyramid** combines (1) deterministic contract tests, (2) LLM-free state/workflow tests, (3) MCP protocol tests, (4) model-in-the-loop golden suites with >= 3 natural paraphrases per core intent and recorded tool traces, and (5) long-dialogue, reload, and compaction tests. Server-verifiable tool traces and pre/post states always supersede model self-reports.

- **Hard gates:** 100 % exact terminal backend state and scope/focus across released Golden Journeys, zero unauthorized mutations, zero cross-session/cross-learner bleed, zero secret/identity leaks, zero regression of global goal/mastery semantics.
- **Quality targets:** >= 3 semantically distinct phrasings per core intent, >= 95 % successful real dialogue runs per scenario, <= 1 demonstrably unnecessary clarifying question, zero repeated already-answered questions, zero visible API jargon, consistent quality across short and long dialogues.

Log field	Manual trial record content
Test ID & Golden Journey	e.g., `GJ-02 / INTENT-003`
Date, environment & versions	Environment (local/staging/prod) + Git SHA, server build, package, and contract version
Learner fixture & launch type	Curriculum, pseudonymized SkillPilot ID, prior state, fresh UI launch / reload / expiry
Verbatim learner inputs	Exact user turns in chronological order
Expected vs. actual tool trace	Expected partial order + forbidden calls vs. telemetry/server-log trace
Backend state & visible rubric	Pre/post state + pedagogical and communicative rubric score
Error class, policies & fix	Affected error class/policy IDs, general root-cause fix (no one-off patch), and retest status

Explicit non-goals: never bend global goal/mastery semantics to paper over chat issues, never add Hesse/Math/LK special-case code, never clone legacy HTTP methods 1:1, never steer behavior via public marketplace blurbs, never trust implicit model memory as durable state, and never claim migration readiness from green schema tests alone.

Definition of Done (10 criteria): (1) all Golden Journeys pass hard gates, (2) every policy ID has closed traceability, (3) multiple curricula and multi-part natural requests work without special-case code, (4) scope and focus resolve before any goal output, (5) teaching, mastery, recall, and exams pass domain review, (6) long dialogues/reloads/compaction/parallel sessions never lose or mix state, (7) one-click launch works with zero manual technical inputs, (8) visible communication meets quality targets, (9) CI, live provider tests, telemetry, and rollback are jointly green, and (10) the legacy Custom GPT coach remains strictly historical reference.

10 SkillPilot: Versioning and Lifecycle Architecture for the OpenAI Plugin

Scope and release status

Binding internal architecture specification governing versioning, dedicated hash-addressed MCP Apps UI resources, bundled skills, and lifecycle management for the OpenAI plugin. The published Git beta `skillpilot-coach-v1` `1.1.2` and its install bytes, together with the rejected `1.0.0` review history (Review Freeze History), remain preserved unchanged. The neutral local `1.1.3` candidate (including the local `9c09821e96ab…` handshake fix alongside retained `12762009…` and deployed `08030…` bytes, with Voice image-sequence host acceptance still pending) represents pre-submission preparation — neither a deployment, portal publication, nor real-client acceptance.

10.1 Decision of 19 October 2026: One Permanent Neutral Plugin

On **19 October 2026**, the Product Owner established a single permanent plugin **SkillPilot Coach** with technical package name **`skillpilot-coach`** and a neutral public MCP origin ahead of directory submission. This supersedes both the initial recommendation to keep `skillpilot-coach-v1` and its endpoint and the earlier one-plugin-per-major identity model.

Boundary	Permanent specification
Technical package name	`skillpilot-coach`
Public display name	`SkillPilot Coach` (<= 30 characters)
Public MCP endpoint & OAuth Resource/Audience	`https://mcp-openai.skillpilot.com/mcp`
Widget origin (`_meta.ui.domain`)	`https://mcp-openai.skillpilot.com`
Protected Resource Metadata	`https://mcp-openai.skillpilot.com/.well-known/oauth-protected-resource/mcp`
Existing OAuth issuer	`https://skillpilot.com/api/openai/v1`
Current internal adapter	`/internal/openai/v1/mcp`, `contractMajor = 1`

Package SemVer, public identity, and domain tool contract are decoupled: future `2.x` package versions may keep the same tool contract, and internal workflow generations require no new public name or origin because OpenAI's plugin update flow does not support changing origins on an already published plugin.

- **Bounded technical cutover & desktop CIMD:** The existing internal V1 adapter, scopes, and session gates stay unchanged (`v1` in the OAuth issuer names the existing authorization server instance rather than constraining package majors). MCP URL, Resource/Audience, widget origin, metadata URL, `/.well-known/openai-apps-challenge`, and trusted edge host move together to `https://mcp-openai.skillpilot.com` — enforcing a single active resource with zero dual-audience acceptance or legacy/new origin dispatchers. Native desktop CIMD uses the exact resource-matched public client `https://chatgpt.com/oauth/codex/7yTcOAOqrumX/client.json` and loopback callback `http://127.0.0.1/callback/7yTcOAOqrumX` (dynamic port, PKCE `S256`), strictly isolated from confidential client authentication with zero fallback.
- **Cutover & rollback (`AuthenticatedClientPolicy`):** Old chats are explicitly excluded (requiring no chat continuation, session migration, or continued legacy MCP runtime, while preserving durable learner progress). Because OAuth resource is part of each client profile

fingerprint, `AuthenticatedClientPolicy` activates the new fingerprint upon startup validation and retires the prior one without rewriting grants (old and new instances must never serve the same database with differing active profiles during cutover). Rolling back to a retired fingerprint is intentionally refused and requires an explicit new `policyRevision`, fresh authorization, and compatibility with all currently approved live tool definitions.

- **Future tool evolution within one identity:** Package metadata, skills, live server, and individual tool definitions update asynchronously on the host; while a revised definition is under review, the previous definition may remain active even in new chats. Incompatible tool changes therefore use distinctly named replacement tools or bounded compatible adapters until live approved definitions allow retiring the old entry (`clientInfo.version`, User-Agent, and prompt prose never drive authorization or contract selection).

10.2 Independent Version Layers and SemVer Classification

No single version identifier may be overloaded across distinct architectural concerns:

Version layer	Example	Meaning & source of truth
Plugin identity	`skillpilot-coach`	Permanent public directory identity and stable MCP origin
Plugin package version	`1.1.3`	SemVer of the submitted bundle (manifest, skills, reviewed MCP/UI metadata)
MCP tool contract major	`1`	Semantics and JSON Schemas of the internal contract adapter (`v1`)
MCP protocol version	date-based	Transport/handshake revision negotiated between host and MCP server
Server build	Git SHA / build #	Deployed backend binary `skillpilot-server`
UI artifact version	`sha256-...`	Immutable content-addressed MCP Apps HTML/JS/CSS bundle
State schema version	`3`	Internal persistence format (decoupled via internal data migrations)
State version (`state Version`)	`142`	Monotonically increasing revision of canonical learner state shared across sessions and WebGUI
Workflow version	`coach@2.3`	Session-pinned pedagogical flow and decision logic
Curriculum revision	`curricula-sha256@...`	Session-pinned learning-goal and skill-graph revision

Draft vs. Published & SemVer transitions:

- **Unpublished working drafts (`DRAFT`):** `contracts/drafts/openai/<plugin>/<version>-SNAPSHOT/` may be regenerated via `prepare` prior to `Submit for Review` (while the manifest carries `<version>` without `-SNAPSHOT`); during active portal review, a review freeze locks package and runtime behavior. `contracts/published/openai/<plugin>/<version>/` and `release-index.json` are created exclusively after confirmed portal publication via the explicit `record-published` command and remain immutable thereafter. Deterministic USTAR archives (`plugin-install-bundle`) and directory submission ZIPs never depend on host `tar` versions, `umask`, checkout permissions, or symlinks, and never bundle the server.
- **Server-only `serverBuild` deploy (zero package bump):** Calculation fixes, performance/database tuning, or internal security hardening that keep the visible auth, tool, schema, skill, and UI contract unchanged (with all contract tests green).

- **PATCH (`x.y.z -> x.y.(z+1)`):** Clarifying tool descriptions, `SKILL.md`, or annotations to match existing behavior, shipping a compatible UI fix under a new hash URI, or updating listing/release notes.
- **MINOR (`x.y.0 -> x.(y+1).0`):** Backward-compatible additions (new tool, optional input field, `extensions` payload, new skill, new UI resource, additive OAuth scope); the live server MUST deploy first as a superset serving both old and new contracts before portal rescanning.
- **Incompatible changes:** Removing/renaming tools or fields, adding required inputs, breaking types/semantics, or turning read-only tools into mutating ones must never deploy unprotected; they require distinctly named replacement tools/superset adapters or an explicit contract-generation transition. Skills ship as a frozen directory tree inside the package (`Skill-Contract-Lint` enforces valid tool references and reproducible bundle hashes).

10.3 Public Contract Adapters, Envelopes, and Dedicated MCP Apps UIs

Public MCP schemas never expose raw internal domain or persistence entities. Stateful responses and write requests carry explicit version and idempotency envelopes (`additionalProperties: false` on core schemas, forward-compatible additions inside `extensions`, structured payloads paired with concise text representations, and zero internal user IDs):

```
{
  "contractMajor": 1,
  "learningSessionId": "sps_...",
  "stateVersion": 42,
  "stateSchemaVersion": 3,
  "workflowVersion": "coach@2.3",
  "curriculumRevision": "curricula-sha256@...",
  "currentStep": { "goalId": "..." },
  "allowedActions": ["submit_answer"],
  "extensions": {}
}
```

Write calls require `learningSessionId`, `expectedStateVersion` (optimistic locking across concurrent transports), and `clientRequestId` (idempotent retries within the `PT1H` and learner-revision gates of 8); pinned `workflowVersion` and `curriculumRevision` values keep in-flight sessions consistent or halt fail-closed with `SESSION_VERSION_UNAVAILABLE`.

Two dedicated V1 MCP Apps UI resources (`text/html;profile=mcp-app`):

1. `render_skillpilot_goal_visualization` (read-only) binds solely its active image-only resource via `ui.resourceUri` and `openai/outputTemplate`, running once per freshly authorized `(goalId, stateVersion)` pair (`expectedStateVersion = stateVersion`) while preserving the complete text-only teaching path.
2. `start_skillpilot_memory_practice` (read-only) binds solely its dedicated interactive flashcard resource.
3. The app-only grading tool `review_skillpilot_memory_practice_card` and all regular domain tools remain unbound (preventing nested or blank UI frames; authoritative state stays strictly in the backend, and corrupted host state falls back to a safe visible error state rather than local learning).
4. **Passive byte retention from first real delivery:** Every content-addressed URI (e.g., `ui://skillpilot/coach/1/sha256-<hash>/index.html`) ever served to a real client remains passively readable with byte-identical content, while tool descriptors bind exclusively the current active URI from `release/line.json`.

10.4 Canonical `contract-line/v1` Schema, CI Gates, Observability, and Line History

Support lifecycle, publication status, and session-launch policy form three orthogonal, server-authoritative axes (never silently toggled by calendar dates):

- **`supportLifecycle`:** `CURRENT` (primary line) | `SUPPORTED` (fully maintained) | `DEPRECATED` (upgrade advised) | `RETIRED` (ended; enforces `UNPUBLISHED` + `BLOCK`).
- **`publicationStatus`:** `DRAFT` | `PUBLISHED` | `UNPUBLISHED`.
- **`newSessionPolicy`:** `ALLOW` (normal launch) | `WARN` (requires a published, verified non-null `successor` on the exact first-party allowlist and explicit user choice) | `BLOCK` (mints zero sessions/capabilities; yields identifier-free `MAJOR_UPGRADE_REQUIRED` handoff when a verified successor exists, or `TEMPORARILY_UNAVAILABLE` when `successor: null`). Monotonic `policyRevision` increments immediately invalidate older launch capabilities.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://skillpilot.com/schemas/openai/contract-line/v1",
  "type": "object",
  "additionalProperties": false,
  "required": [
    "contractMajor",
    "policyRevision",
    "displayName",
    "supportLifecycle",
    "publicationStatus",
    "newSessionPolicy",
    "successor"
  ]
}
```

(Full property constraints and `allOf` invariants enforcing `WARN -> successor != null` and `RETIRED -> UNPUBLISHED + BLOCK` are validated in release and CI checks against `$id`; following the Web-first cutover, V1 stands at `policyRevision: 3`, `CURRENT`, `DRAFT`, `ALLOW`, `successor: null`.)

- **Mandatory CI/CD gates:** (1) version, naming, and manifest-limit consistency, (2) automated `Contract-Diff` against published snapshots (blocking removed tools/fields, new required inputs, type/enum/security breaks, and mutated immutable UI URIs), (3) `Skill-Contract-Lint` and deterministic bundle hashes, (4) `Contract-Testmatrix` across all non-`RETIRED` lines including OAuth audience/scope and UI integrity checks, and (5) state-migration and rollback tests.
- **Observability & log privacy:** Every tool invocation records `contractMajor`, `pluginLine`, `serverBuild`, `toolName`, `status`, stable `resultCode`, latency, and (when present in versioned responses) `stateVersion`, `stateSchemaVersion`, `workflowVersion`, `curriculumRevision`, plus `clientRequestId` on writes. `learningSessionId` is logged exclusively as a truncated HMAC-SHA-256 fingerprint and never in plaintext; high-cardinality correlators and revisions remain log fields only, while metric tags are strictly bounded to `contractMajor`, `pluginLine`, `toolName`, `status`, and `resultCode`.
- **Historical record of the superseded multi-plugin-per-major model:** The July–September 2026 plan envisioned shipping removable breaking major contracts as separate directory identities (`skillpilot-coach-v1`..`v9` on reserved `mcp-coach-v1`..`v9.skillpilot.com` hosts, cf. the 31 July 2026 OpenAI Developer Forum inquiry) with parallel operation, optional explicit `mig_...` session migration, and eventual unpublish/delete, while already retiring unshipped locale-split hosts (`mcp-coach-de-v*`/`en-v*`). Under the 19 October 2026 decision (10.1), clean internal adapter separation (`v1`, and future `v2` inside the same `skillpilot-`

`server` when needed) is preserved while maintaining a single permanent external plugin identity (`skillpilot-coach` on `https://mcp-openai.skillpilot.com/mcp`).