Nick Clark · Published April 25, 2026 · PDF

How to License AI Agent Skills That Only Unlock When Credentials Intersect

You want an agent skill to become usable only when several independent conditions are satisfied at once: a proven competency, a bound external credential, and a policy scope that matches the deployment. Static role checks and API keys cannot express that intersection, and they cannot revoke access when one factor later fails. This guide walks through an architecture for composite, evidence-backed skill licensing disclosed in United States Patent Application 19/647,395, whose home inventive step is the LLM and Skill Gating inventive step. It is an architecture you implement yourself, not a library you can install.

What You Are Building

You are building a licensing layer for AI agent skills where a skill does not become exercisable because a caller holds a token or sits in the right role. It becomes exercisable only when a set of independent conditions all hold at the same moment: the requester has demonstrated the underlying competency, an external credential has been bound to that same requester, and the governing policy scope matches the system the skill will run on. When any one of those factors lapses, the skill closes again.

This is the problem developers hit when a capability carries real consequence. A repair agent that can command torque on a machine, a financial agent that can move funds, a clinical assistant that can surface protected data: for each, "the API key was valid" is not a defensible reason for the action to have happened. What you want is a gate that treats access as the intersection of several kinds of proof, evaluates it continuously, and produces an auditable record of why the skill opened.

The approach described here is disclosed in United States Patent Application 19/647,395. It is an architecture, not a shipping product or a downloadable SDK. You build the components yourself against your own runtime.

Why the Obvious Approaches Fall Short

The usual way to license a capability is to issue a credential that asserts something and then check for its presence: an API key, an OAuth scope, a role in an access-control list, a signed JWT with claims. These mechanisms are real, widely deployed, and genuinely good at what they do, which is answering "does the caller possess this token." The structural gap is what they answer.

A token attests to a past event: an account was provisioned, a role was granted, a course was completed. It does not measure present competence, and it does not know whether the entity presenting it is the entity it was issued to. The application described in 19/647,395 makes this point directly about static credentials: passwords, tokens, certificates, and biometric templates "assert identity at discrete points in time" and cannot distinguish an authorized individual from an unauthorized one who holds that individual's credentials.

Two further gaps matter for skills that carry consequence. First, conventional credentials are granted once and assumed to hold; there is no built-in path for access to close when performance later degrades. Second, they are typically single-factor in the sense that matters here: possessing the token is sufficient. Expressing "this skill opens

only when competency AND a bound credential AND a matching policy scope all hold" usually devolves into ad hoc glue code scattered across the call site, with no shared evaluation point and no audit trail explaining the decision.

The Architecture

The disclosed approach replaces the token check with a governed evaluation point and makes access the intersection of independently established factors. Every mechanism below traces to the filing.

The capability gate. At the center is what the application calls an evidence-based capability gate: "a governed evaluation point that stands between a requester and a capability that the requester seeks to exercise." Crucially, it does not rely on "credentials that attest to past training, degrees that attest to past education, or role assignments that attest to organizational position." It evaluates demonstrated performance evidence against defined competency thresholds and produces one of two outcomes: progressive unlock, or regression and revocation. The gate is described as a continuous evaluation, not a one-time assessment: it can close and revoke a previously granted capability "if the requester's ongoing performance evidence indicates that competence has degraded below the required threshold."

Where the evidence comes from. Feeding the gate is a curriculum engine that "produces mastery evidence through structured assessment and continuous operational monitoring." In the filing the curriculum engine defines, for each gated capability, a set of learning objectives, assessment instruments, a sequencing policy, and a mastery threshold per objective. It implements progressive unlock: capabilities "are unlocked progressively as the requester demonstrates mastery of increasingly complex or critical aspects," rather than in a single pass-or-fail event. The curriculum itself is a governed object whose modifications are validated, policy-checked, and recorded in lineage, so a curriculum cannot be quietly weakened to lower the bar.

The certification token as the license. When the gate opens, the system "generates a certification token." The filing is explicit that this is not a conventional credential: "not a role assignment, a permission grant, or a static badge," but "a time-bounded, evidence-backed, cryptographically verifiable attestation that is subject to expiration, revocation, and revalidation." Its fields are the load-bearing part of the design. Per the application the token carries: a capability identifier; the holder identity; an evidence hash (a cryptographic hash of the evaluated evidence corpus, letting a verifier confirm what the token was issued against without seeing the evidence); issuance and expiration timestamps; the policy scope under which it was issued; the issuing authority; a device entropy binding "to the physical device from which the mastery evidence was submitted, preventing token portability to devices on which the mastery was not demonstrated"; and the issuing authority's signature.

Where the intersection lives. The credential-intersection behavior you are after is assembled from three disclosed mechanisms that compose. The application's compositional binding module "evaluates policy-governed binding requirements that may demand single-substrate, dual-substrate, or tri-substrate identity validation depending on the action's governance requirements." Its credential binding module binds an external credential (a badge, a professional certification, a government document) to a validated identity through a binding event, so that later "a genuine passport presented by an individual whose biology does not continue the bound trust-slope is rejected." And its multi-identity authorization module evaluates policies "requiring authorization from multiple independent biological identities before a resource action is permitted," for example a two-person requirement, "evaluating each trust-slope independently without cross-disclosure." Composite licensing is the conjunction: the skill opens only when the competency gate is open AND the required credential binding verifies AND the compositional or multi-identity policy is satisfied AND the token's policy scope matches.

Cross-platform deployment gating. The filing also describes what happens when the license is presented to a system other than the one that issued it. A deployment gate "verifies the token's cryptographic signature against the issuing authority's public key, validates the token's expiration status, and evaluates the token's policy scope for compatibility with the receiving system's own governance requirements." Acceptance is "subject to any additional requirements imposed by the receiving system's own capability gate." The receiving side never trusts the token blindly; it re-checks scope and can layer its own gate on top.

Why a language model cannot open its own gate. Because these systems often use a language model to propose actions, the application places the model outside the trust boundary. Under its structural-starvation approach the model is "denied access to the informational resources that would be necessary for hallucination to occur," and if the multimodal pipeline detects gaming, "the trust weight assigned to language model proposals that reference the compromised evidence is reduced," so a proposed unlock built on flagged evidence is preferred against or rejected. The gate decision is made by the governed evaluation point, not by the model requesting the skill.

How to Approach the Build

The following order lets you stand up the intersection incrementally. The sketches are illustrative and faithful to the filing; they are not runnable code.

1. **Define the gated skill and its competency thresholds.** For each skill you intend to license, name the capability with a stable identifier and, following the curriculum-engine model, enumerate the objectives and the mastery threshold each must clear. Keep this definition as a governed object with its own change history so nobody can silently lower a threshold.
2. **Build the evidence source, then the gate as a single evaluation point.** Route every unlock request through one capability gate rather than checking conditions at each call site. Conceptually:

```
decision = capability_gate.evaluate(
    requester,
    capability_id,
    evidence = curriculum.mastery_evidence(requester, capability_id),
    policy)
# decision -> PROGRESSIVE_UNLOCK | REGRESSION_REVOKE   (illustrative)
```

The gate compares accumulated evidence to the thresholds and returns unlock or revoke. Because the filing describes this as continuous, plan for the gate to be re-run on ongoing operational evidence, not only at first request.

3. **Express the intersection in policy, not in glue code.** Encode the composite requirement as a policy the gate reads: which competency threshold, which external credential must be bound, whether single-, dual-, or tri-factor binding is required, and any multi-identity (for example two-person) requirement. The skill opens only when the conjunction holds. Keeping this in policy is what lets you audit and change the intersection without touching the skill.
4. **Bind external credentials to the requester before you trust them.** Implement the binding event: capture the credential and the requester identity together and record the association. On later use, re-verify that the presenter is the bound owner, and fail the credential if that continuity check fails even when the credential artifact itself is genuine.
5. **Issue a certification token with all disclosed fields.** On unlock, mint a signed token carrying capability id, holder, evidence hash, issuance and expiration timestamps, policy scope, issuing authority, and the device entropy binding. Treat the token as the license the skill loader checks, and honor the full lifecycle: active, expired, revoked, revalidated, with each transition recorded in lineage.

6. **Gate the actual skill load on token verification and scope match.** Only when a valid, unexpired, unrevoked token is presented and its policy scope matches the running system should the scoped skill or adapter be made exercisable. On a receiving system, run the deployment-gate checks (signature, expiration, scope compatibility) and layer your own capability gate on top before honoring an externally issued token.
7. **Wire revocation and revalidation as first-class events.** Because access is meant to close, connect your operational monitoring back to the gate so degraded evidence, an expired token, or a governance action revokes the license and forces revalidation before the skill reopens.

What This Does Not Give You

This is an architecture, not a drop-in library, and there is no package to install. You implement every component yourself against your own runtime, identity substrate, and policy engine, and the integration work is substantial: an evidence pipeline, a governed curriculum store, a signing and revocation infrastructure, and a policy layer that expresses the intersection.

The application discloses the method; it does not report benchmarks, latency figures, or production deployment data, and nothing here should be read as a performance guarantee. Whether the intersection is meaningful depends entirely on the quality of the evidence you collect and the honesty of your thresholds. A weak assessment or a rubber-stamp credential binding produces a gate that looks rigorous and is not.

The disclosed identity and evidence mechanisms lean on continuous, sometimes multimodal signals; if your context cannot produce trustworthy competency evidence, or if a simple static token is genuinely adequate for the risk involved, this architecture is heavier than the problem warrants. Use it where the consequence of a wrongful unlock justifies the cost, not everywhere.

Disclosure Scope

The architecture described in this guide is disclosed in United States Patent Application 19/647,395. This guide is educational: it explains an approach to composite, evidence-backed skill licensing so that a developer can understand and build it, and it describes only what that filing discloses. It is not a warranty, a specification of a shipping product, or an offer of software, and no downloadable library or SDK is provided or implied. Any third-party technologies mentioned for context are referenced neutrally and are the property of their respective owners.

LLM & Skill Gating (/llm-skill-gating)

All 40 steps → (/inventive-steps)

The model proposes. The agent decides.

Chapter 7 (/patents/19-647395/chapters/llm-skill-gating)

PRIMARY TECHNICAL DISCLOSURE

- AI-Mediated Curriculum and Progressive Capability Unlocking Using Semantic Performance States (/articles/ai-mediated-curriculum-and-progressive-capability-unlocking-using-semantic-performance-states)

SECONDARY TECHNICAL

- LLM as Structurally Untrusted Proposal Generator (/articles/llm-skill-gating/untrusted-proposals)
- Mutation-Validation-Arbitration Pipeline (/articles/llm-skill-gating/mutation-validation-pipeline)
- Hallucination Prevention via Structural Starvation (/articles/llm-skill-gating/structural-starvation)
- Trust Weight Calibration and Decay (/articles/llm-skill-gating/trust-weight-calibration)
- Evidence-Based Capability Gating (/articles/llm-skill-gating/evidence-gating)
- Certification Token Generation (/articles/llm-skill-gating/certification-tokens)
- Narrative State and Personality Architecture (/articles/llm-skill-gating/narrative-personality)
- Skill Regression Detection and Capability Revocation (/articles/llm-skill-gating/regression-detection)
- Arbitration as Semantic Event (/articles/llm-skill-gating/arbitration-events)

- Structural Starvation as a Composable Safety Primitive (/articles/llm-skill-gating/starvation-composability)
- Multi-Turn Memory Isolation (/articles/llm-skill-gating/memory-isolation)
- Curriculum Engine Progressive Unlock (/articles/llm-skill-gating/curriculum-engine)
- Multimodal Evaluation Pipeline (/articles/llm-skill-gating/multimodal-evaluation)
- Multimodal Anti-Gaming Substrate (/articles/llm-skill-gating/anti-gaming)
- Professional Skill Gating Applications (/articles/llm-skill-gating/professional-gating)
- Embodied Skill Gating (/articles/llm-skill-gating/embodied-gating)
- Biological Identity Skill Binding (/articles/llm-skill-gating/biological-binding)
- Security and Drift Detection Layer (/articles/llm-skill-gating/security-layer)
- Validation Feedback Asymmetry (/articles/llm-skill-gating/feedback-asymmetry)

APPLICATIONS · GENERAL

- Progressive AI Agent Deployment: Granting Authority Through Earned, Continuously-Evidenced Capability (/articles/llm-skill-gating/enterprise-progressive-deployment)
- Educational Platform Competency Through Structural Certification (/articles/llm-skill-gating/educational-competency)
- How to License Medical AI: Evidence-Gated Clinical Capability and Competence Governance (/articles/llm-skill-gating/medical-licensing)
- Jurisdiction-Gated Legal AI: Certifying Practice-Area Competence Before an LLM Gives Advice (/articles/llm-skill-gating/legal-practice-certification)
- AI Flight Training That Gates Pilot Privileges on Demonstrated Competence, Not Credentials (/articles/llm-skill-gating/aviation-pilot-training)
- AI Financial Advisor Certification: Fiduciary-Grade Skill Gating for Investment Advice (/articles/llm-skill-gating/financial-advisor-certification)
- Skill Gating for Cybersecurity AI: Earning Dangerous Capabilities Through Evidence (/articles/llm-skill-gating/cybersecurity-skill-progression)
- LLM and Skill Gating for Manufacturing Quality Systems (/articles/llm-skill-gating/manufacturing-quality)
- Decentralized Agent Skill Marketplace: Cryptographic Skill-to-Authority Binding for AI Agent Platforms (/articles/llm-skill-gating/agent-skill-marketplace)
- Runtime LoRA Adapter Admission Control for Regulated AI Deployment (/articles/llm-skill-gating/runtime-lora-loading)
- Multi-Authority AI Licensing Compliance at the Inference Boundary (/articles/llm-skill-gating/composite-licensing-intersection)

APPLICATIONS · SPECIFIC

- Duolingo Alternative: Evidence-Gated Capability vs Content Unlock (/articles/llm-skill-gating/duolingo)
- Khan Academy Khanmigo vs Evidence-Gated Capability Unlock (/articles/llm-skill-gating/khan-academy)
- Coursera Alternative for Competence-Verified Credentials: Governed Skill Gating (/articles/llm-skill-gating/coursera)
- GitHub Copilot vs Evidence-Gated Code Generation (/articles/llm-skill-gating/github-copilot)
- Pearson Alternative for Governed Capability Progression: Evidence-Gated Skill Unlocking (/articles/llm-skill-gating/pearson)
- Chegg vs Evidence-Gated Capability Unlock: A Governed Alternative to Answer Access (/articles/llm-skill-gating/chegg)
- Grammarly Alternative for Evidence-Gated Writing Capability (/articles/llm-skill-gating/grammarly)
- Is there a Photomath alternative that makes students earn each solution? (/articles/llm-skill-gating/photomath)
- Century Tech Alternative: Evidence-Gated Mastery Beyond Adaptive Recommendation (/articles/llm-skill-gating/century-tech)
- Squirrel AI vs evidence-based capability gating: which certifies mastery? (/articles/llm-skill-gating/squirrel-ai)
- Anthropic Skills Alternative: Consumer-Side Certification for Governed Skill Admission (/articles/llm-skill-gating/anthropic-skills)
- OpenAI Custom Actions Alternative: Evidence-Gated Action Authority (/articles/llm-skill-gating/openai-custom-actions)
- Google Gemini Extensions vs Evidence-Gated Capability Unlocking (/articles/llm-skill-gating/google-gemini-extensions)
- Microsoft Copilot Studio Alternative for Sovereign and Air-Gapped Agent Governance (/articles/llm-skill-gating/microsoft-copilot-studio)
- HuggingFace PEFT vs Evidence-Gated Capability: Governed Skill Activation Beyond Adapter Loading (/articles/llm-skill-gating/huggingface-peft)
- Meta Llama Guard vs Governed Skill Gating: Content Filtering Beyond the Safety Classifier (/articles/llm-skill-gating/meta-llama-llama-guard)
- OpenAI Operator vs Evidence-Gated Agent Execution (/articles/llm-skill-gating/openai-gpt4o-operator)
- Windsurf Alternative: Evidence-Gated Agent Capability Beyond Workspace Toggles (/articles/llm-skill-gating/codeium-windsurf)

LLM & Skill Gating overview → (/llm-skill-gating)