Nick Clark · Published July 2, 2026 · PDF

How to Gate AI Agent Skills Behind Competency Checks

If you build agents that can call powerful tools, you eventually face a hard question: how do you let an agent (or a person) earn access to a capability by demonstrating it, instead of by holding a static token or role? This guide walks through an architecture for competency-gated, progressively unlocked skills, grounded in the disclosure of United States Patent Application 19/647,395. It describes the LLM and Skill Gating inventive step as an approach you build yourself, not as a shipping library you install.

What You Are Building

You are building a control point that decides whether a requester may exercise a given skill, based on accumulated evidence that the requester can exercise it competently, rather than on a credential, role, or static permission grant. The requester might be a human operator, a semantic agent, or a composite system. The skill might be a dangerous tool call, a privileged action, or a scoped capability module that should only load once the requester has earned it.

The search-intent problem is concrete. You have agents that can do real damage if they act beyond their demonstrated ability, and you want access to expand as competence is demonstrated and contract when it degrades. A pass/fail exam at onboarding does not solve this, because competence is not a permanent property. This guide describes an

evidence-based capability gate, a curriculum engine that feeds it, a progressive-unlock model, and a certification-token lifecycle, all as disclosed in United States Patent Application 19/647,395.

Why the Obvious Approaches Fall Short

The conventional pattern is role-based or credential-based access control. A requester holds a role, a token, or a certificate, and the system checks for its presence. This is the model behind most permission systems, and it works well for its intended purpose: attesting that some authority once granted something.

The structural gap is that these mechanisms attest to the past. A degree attests to past education, a role assignment attests to organizational position, and a static token attests that a check passed at issuance time. None of them measures whether the holder can competently exercise the capability in the current context. Once granted, access does not contract when performance degrades, because nothing is watching performance after the grant. The disclosure calls this out directly: the capability gate it describes deliberately does not rely on credentials, degrees, or role assignments, because those attest to the past rather than to demonstrated present ability.

A second, subtler gap appears once a language model is in the loop. If you let an LLM decide when to unlock a skill, you have made the model the decision-maker. The disclosure takes the opposite stance: every language-model output is a proposal, never an authoritative decision, and no proposal reaches a capability gate, a token, or any agent state without passing through an agent-resident validation path. A gate that trusts the model to grade the requester can be talked into opening.

The Architecture

The disclosed architecture has four parts. Everything below traces to the filing.

The capability gate. A capability gate is a governed evaluation point standing between a requester and a capability the requester seeks to exercise. It evaluates the requester's accumulated evidence of competence in the relevant domain and produces a binary determination: open (grant access) or closed (deny access). The critical property is that it is a continuous evaluation, not a one-time assessment. If ongoing performance evidence shows competence has fallen below the required threshold, the gate can close and revoke a capability it previously granted. Evidence comes from two sources: structured assessment through the curriculum engine, and continuous operational monitoring of performance after the capability was granted.

The curriculum engine and progressive unlock. The curriculum engine defines, sequences, and administers the activities through which a requester accumulates performance evidence. For each gated capability it defines learning objectives, assessment instruments, a sequencing policy, and a mastery threshold per objective. Rather than granting a capability in a single event, it implements progressive unlock: the requester is exposed to simpler and lower-risk aspects of a capability first, and access to more complex or higher-risk aspects opens only as mastery of the earlier aspects is demonstrated. In the disclosure each curriculum is itself a governed object: changes to objectives, thresholds, or sequencing are governed mutations that are validated, policy-checked, and recorded in the curriculum's lineage, so a curriculum cannot be quietly weakened or bypassed without an attributable, auditable policy change.

The certification token and its lifecycle. When a gate opens, the system generates a certification token: a cryptographically signed object attesting to demonstrated mastery of a specific capability, at a specific time, under specific assessment conditions. The disclosure is explicit that this is not a conventional badge or role grant. It is a time-bounded, evidence-backed attestation. Its fields include a capability identifier, the holder's identity, an evidence hash (a hash of the evaluated evidence corpus, so a verifier can confirm what the token was issued against without seeing the raw evidence), issuance and expiration timestamps, the policy scope, the issuing authority,

a device-entropy binding to the device from which the mastery evidence was submitted, and the issuer's signature. The token moves through a defined lifecycle: active, expired (validity window elapsed), revoked (invalidated by mastery regression, incident reports, or governance intervention, regardless of expiration), and revalidated (a fresh token issued after successful re-assessment). Each transition is recorded as a governed event in the holder's lineage. A revalidated token can feed a deployment gate that lets another platform accept it, after checking the signature, expiration, and policy-scope compatibility against that platform's own gate.

The multimodal evaluation pipeline as anti-gaming substrate. The evidence that feeds the gate comes from a pipeline that can ingest multiple modalities (text, audio, video, sensor telemetry, biometrics), each producing an independent score vector fused into a composite. The disclosure gives this a second job beyond richer scoring: detecting gaming. It names four mechanisms. Cross-modality consistency enforcement flags cases where one signal claims mastery while another contradicts it (for example, expert text output alongside physiological markers of overload consistent with reading from an external source). Temporal pattern analysis looks for response-timing signatures of coaching or automated generation. Spoofing detection uses continuous identity verification plus behavioral-biometric continuity to catch mid-session substitution. And LLM proposal down-weighting reduces the trust weight of any model proposal that references evidence the anti-gaming substrate has flagged, so a flagged unlock proposal loses to alternatives or is rejected.

One design principle ties these together and is worth stating on its own. The disclosure keeps capability and permission in architecturally separate subsystems with no bidirectional dependency, combined only at an execution gate where both must be satisfied. Competence does not make a requester more authorized, and authorization does not make a requester more competent. A skill gate should sit at that intersection.

How to Approach the Build

You are implementing this yourself. The steps below are the order the architecture implies.

1. **Model each gated capability as an object, not a boolean.** Give it a capability identifier, a set of learning objectives, a mastery threshold per objective, and a sequencing policy. Treat this definition as governed: changes should be validated and recorded, not edited in place with no trail.
2. **Define what counts as evidence, per modality.** For each objective, decide which signals attest to it and how each is scored into a per-dimension vector. Then define the fusion rule that produces a composite, and decide what inter-modality disagreement means. Do not average away contradictions; the disclosure treats a text-versus-physiology conflict as signal, not noise.
3. **Build the gate as a threshold over accumulated evidence.** An illustrative interface sketch, faithful to the disclosure and not a working library:

   ```
   # Illustrative only. You implement the internals.
   evaluate_gate(requester, capability):
       evidence = pipeline.composite(requester, capability)   # fused, multi
       if anti_gaming.flagged(evidence): down_weight(evidence)
       return evidence.meets(capability.thresholds)           # open or clos
   ```

   The gate returns open or closed. Keep it re-runnable, because it must also run after a grant.
4. **Implement progressive unlock as staged thresholds.** Order objectives from simpler and lower-risk to more complex and higher-risk, and only expose a higher stage once the earlier stages' evidence clears their thresholds.

5. **Issue a certification token when the gate opens.** Populate the disclosed fields, including the evidence hash, expiration, policy scope, device-entropy binding, and issuer signature. Record issuance in the holder's lineage.
6. **Run the lifecycle, including revocation.** Wire continuous operational monitoring back into the gate so degraded performance can move a token to revoked. Support expiration and a revalidation path that issues a fresh token after re-assessment. This closing loop is the part that role-based systems lack, so do not skip it.
7. **Keep any LLM on the proposal side of the boundary.** If a model suggests unlocking a skill, route that suggestion through your own validation before it can affect the gate or mint a token. There should be no path by which model output becomes an authoritative grant.
8. **Keep capability and permission separate.** Evaluate "can they do it" and "are they allowed to do it" independently, and require both at the execution point.

What This Does Not Give You

This is an architecture, not a drop-in library. There is no package to install and nothing here "just works" out of the box. The pseudocode above is illustrative; you write the pipeline, the fusion rules, the token format, the lineage store, and the monitoring loop yourself.

The approach is disclosed in a patent filing. It has not been presented here as a shipping product, and this guide states no benchmarks, latencies, accuracy figures, or production results, because the disclosure states none and you should not either. The anti-gaming mechanisms are described as detection and down-weighting measures; the disclosure does not claim they are unbeatable, and cross-modality checks in particular depend on your having trustworthy modalities to compare. The device-entropy binding and continuous identity verification reduce portability and substitution risk but rest on your identity and device-attestation choices, which are out of scope here.

The architecture is a fit where competence is demonstrable, degradable, and worth continuously re-checking, and where the cost of over-granting is high. It is overkill where a simple static role is genuinely sufficient, and it cannot manufacture reliable evidence from modalities you do not actually collect.

Disclosure Scope

The approach described in this guide, including the evidence-based capability gate, the curriculum engine and progressive unlock model, the certification-token lifecycle, and the multimodal anti-gaming substrate, is disclosed in United States Patent Application 19/647,395. This guide is educational. It is provided to teach the architecture and how a developer might approach building it, and it is not a warranty, a specification, or an offer of software. Nothing here should be read as a claim that a productized, benchmarked, or downloadable implementation is being distributed.

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)