GLYPHD LABS

WORKING PAPER 29

Disjointed, Yet Joined

The Ambient Integration Web and the human operator in a world of self-adapting independent projects

Hayden Lindley / Glyphd Labs

2026-07-30 / Version 0.1.0

EVIDENCE: HYPOTHESIS

Contents

01 Abstract

The web already contains many of the mechanisms required for software to notice change and adapt: feeds announce new information; publish-subscribe protocols distribute updates; machine-readable descriptions expose interfaces; dependency bots propose upgrades; controllers reconcile current state toward declared state; agent protocols allow independently built systems to discover capabilities; authorization and supply-chain frameworks constrain access and preserve evidence. These mechanisms are real, but they remain disjointed. They usually operate inside a known dependency graph, a single administrative domain, or a prearranged integration.

This paper proposes the Ambient Integration Web: a future web in which independently owned projects maintain machine-readable missions, constraints, capabilities, and acceptance policies, while bounded project scouts search the public technological environment for changes that could improve them. A capability published by one organization could be recognized, tested, and proposed as an upgrade by another project without a formal partnership or prior bilateral relationship. The systems remain institutionally separate while becoming functionally joined.

The paper first traces the existing lineage from syndication, web-scale publish-subscribe, API description, dependency automation, control loops, federated protocols, agent interoperability, authorization, and provenance. It then specifies a project-level architecture built from Capability Beacons, Project Constitutions, Scouts, Fit Assessments, Adapter Forges, Verification Packets, human-governed proposal inboxes, and receipts. The central safety claim is that ambient discovery must not imply ambient authority: agents may discover and construct candidate changes, but canonical project state changes only through scoped authorization, verification, acceptance, and reversible publication.

The final proposition concerns the human operator. As machine systems assume more search, synthesis, testing, and maintenance, the operator's primary role moves upward: from manually finding every connection to authoring purpose, setting boundaries, allocating authority, judging exceptions, accepting consequences, and remaining accountable to the people affected. The machine can increasingly answer what can be connected and how. The human remains responsible for whether it should be, for whom, and under what conditions.

02 1. Problem and motivation

The contemporary web connects documents, services, packages, identities, devices, repositories, and people. Yet most meaningful integration still depends on a human noticing that two things belong together.

A developer reads an announcement. A product manager imagines a use. Two organizations arrange a partnership. Engineers study documentation. Someone writes an adapter. Tests are created. Permissions are negotiated. A change is released. The web may provide every transport and interface required for this work, but the recognition of relevance remains largely manual and episodic.

This creates an important asymmetry. Capabilities can now be published globally in minutes, but their useful incorporation remains bounded by the attention of people who already know to look. A small project may have a clear need that a large platform has just made solvable, yet neither side knows the other exists. The large platform cannot enumerate all downstream opportunities. The small project cannot continuously inspect every relevant standards body, vendor release, research result, public dataset, model, regulation, grant, security advisory, and neighboring project.

The missed connection is not necessarily a missing API. It is a missing project-level recognition loop.

Consider a hypothetical independent service, Possum Party, whose application helps professional possum trappers work more safely and humanely. A large platform publishes an Animal Keep API and SDK with new capabilities for stress-signal classification, geofenced release records, and humane-handling compliance. Possum Party is not a customer selected for a partnership. The platform does not own it. No employee at either organization initiates contact.

Still, Possum Party's project scout may recognize that the new capability advances three accepted goals, conflicts with one privacy constraint, and could replace an unreliable local component. It can construct a bounded adapter, run a fixture suite, estimate operating cost, identify required permissions, and place a proposal before the human operator.

The two institutions remain separate. Their systems have nevertheless joined themselves functionally.

This paper calls that condition disjointed, yet joined.

03 2. Thesis and contribution

The central thesis is:

> *The next interconnected web will not be formed only by people linking pages or companies signing integrations. It will also be formed by independently governed projects that continuously discover public capabilities, evaluate them against their own purpose, and propose evidence-bearing changes to themselves.*

The proposed name is the Ambient Integration Web.

"Ambient" does not mean invisible authority or uncontrolled background mutation. It means that potentially relevant capability is continuously present in the environment, just as published information is continuously present to search engines today.

"Integration" means a tested functional relationship, not necessarily a commercial partnership, shared ownership, or permanent dependency.

"Web" means the public, addressable environment through which projects can publish capability, identity, constraints, events, evidence, and machine-readable terms.

The specific contribution is not a new transport protocol. Most transports already exist. The contribution is a project-level composition that connects:

1. public capability emission;

2. mission-aware opportunity discovery;
3. bounded candidate construction;
4. evidence and provenance;
5. scoped authority;
6. human acceptance;
7. reversible change;
8. ecosystem feedback.

The paper also distinguishes this proposal from ordinary dependency automation. A dependency bot asks whether a declared package has a newer version. An Ambient Integration scout asks whether something not currently inside the dependency graph could advance the project's accepted mission without violating its constitution.

That is a substantially wider search problem.

04

3. The automated web already exists in fragments

The Ambient Integration Web should not be presented as a clean break from the existing internet. Its component behaviors have been emerging for decades. What is new is their possible composition around independent project purpose.

3.1 Feeds made change observable

Atom standardized a machine-readable format for web feeds and entries. Its primary use case was content syndication, but the deeper architectural move was to let software observe that a public source had changed without requiring a person to revisit the page.

A feed does not understand whether an entry matters to a project. It does make public change enumerable, timestamped, attributable, and retrievable.

This is the first fragment: the web can emit legible change.

3.2 Publish-subscribe made change pushable

Polling is wasteful when updates are sparse. WebSub formalized a decentralized HTTP-based publish-subscribe pattern in which publishers advertise hubs and subscribers receive new content through them.

CloudEvents addressed a neighboring problem: event producers describe occurrences differently, limiting interoperability among services and infrastructure. Its common event envelope allows independently developed producers and consumers to exchange event context in a more portable form.

These systems still do not decide what an event means to an arbitrary project. They establish the second fragment: the web can distribute change to interested machines.

3.3 Interface descriptions made capability inspectable

OpenAPI gives humans and computers a language-agnostic description of an HTTP service. W3C Web of Things Thing Descriptions similarly describe metadata, interaction affordances, data schemas, security definitions, and links for physical or virtual entities.

A machine can therefore inspect more than a product announcement. It can learn that a service exposes an action, what inputs it expects, what data it returns, and what security scheme protects it.

These descriptions are generally technical. They say how a capability can be invoked. They rarely say which projects should care, what social purpose the capability serves, or which tradeoffs it introduces.

This is the third fragment: the web can describe what a system can do.

3.4 Dependency bots made update proposals routine

Package managers created explicit dependency graphs. Dependabot and related systems inspect those graphs, identify outdated or vulnerable dependencies, and raise pull requests containing proposed updates. A human can review the diff, changelog, and tests before merging.

This is already a small version of the future described here:

- a project declares part of its machine-readable state;
- an automated system monitors an external ecosystem;
- a relevant change is detected;
- a patch is constructed;
- the patch arrives as a proposal rather than silently becoming truth.

Its limitation is equally instructive. The system operates primarily inside dependencies the project already knows it has. It does not normally discover that an unrelated capability should become a new part of the project.

This is the fourth fragment: software can propose maintenance changes to software.

3.5 Controllers made desired-state reconciliation continuous

Kubernetes controllers watch current state and attempt to move it closer to declared desired state. The operator pattern extends this by encoding application-specific operational knowledge that a human operator would otherwise apply manually.

This architecture demonstrates that a system does not need a single terminal task with a final static answer. It can continuously compare what exists with what ought to exist and take bounded actions to reduce the difference.

The Ambient Integration Web generalizes this logic beyond a cluster. Instead of reconciling only resources already represented in one control plane, a project scout observes a public capability environment and asks whether the desired state itself has become newly achievable.

This is the fifth fragment: software can continuously reduce a declared state gap.

3.6 Federation made cooperation possible without common ownership

ActivityPub provides decentralized client-to-server and server-to-server APIs for content and notification exchange. Federation proves an important institutional point: independently administered systems can participate in a shared protocol without belonging to one company or database.

The protocol does not require every server to expose its internal memory or implementation. It establishes sufficient common form at the boundary.

This is the sixth fragment: separate institutions can interoperate through public contracts.

3.7 Agent protocols made capability discovery explicit

Model Context Protocol standardizes how language-model applications connect to external data and tools. Agent2Agent targets communication among independent, potentially opaque agent systems, including capability discovery, interaction negotiation, and collaborative task management.

These protocols move the web closer to active machine participants. Yet tool access or agent communication is not equivalent to project evolution. A project still needs an authoritative statement of what it is, what it wants, what may be changed, and what evidence is required.

This is the seventh fragment: machine actors can discover and invoke one another's capabilities.

3.8 Authorization and provenance made bounded trust representable

OAuth separates a client from a resource owner and allows limited access rather than surrendering the owner's full credentials. in-toto lets a project owner define an authorized software-supply-chain layout, record evidence about steps and artifacts, and verify that the resulting chain followed the declared plan.

These mechanisms are crucial because ambient discovery creates ambient attack surface. The ability to find a capability must never imply permission to use it. The ability to generate a patch must never imply authority to merge it. The ability to execute a workflow must never imply that its outputs are trusted.

This is the eighth fragment: authority and evidence can be scoped, recorded, and verified.

05 4. What is still missing

The fragments above are substantial, but they do not yet form an Ambient Integration Web.

Five missing objects prevent the composition.

4.1 A project does not usually publish its purpose as operable state

A repository may expose a README, roadmap, issue tracker, API, package manifest, or policy file. These artifacts are not normally compiled into a machine-operable statement of:

- mission;
- intended beneficiaries;
- accepted goals;

- current constraints;
- non-goals;
- risk tolerances;
- privacy boundaries;
- legal obligations;
- budget;
- current capabilities;
- unresolved needs;
- acceptance criteria;
- authority owners.

Without this state, a scout can match keywords but cannot reliably judge fit.

4.2 Capability descriptions omit consequential meaning

An API description can expose endpoints and schemas. It seldom states:

- why the capability exists;
- which classes of project it may help;
- known harmful uses;
- data-retention behavior;
- expected cost;
- jurisdictional restrictions;
- model limitations;
- provenance of outputs;
- compatibility promises;
- deprecation horizon;
- evidence required for responsible use.

A capability is more than a callable interface. It is a possible alteration to another project's behavior.

4.3 Automated update systems search known graphs

Dependency automation is powerful because the graph is explicit. The package is already declared. The desired change is narrow: update the version while preserving contracts.

Ambient integration requires open-world opportunity discovery. The scout must consider capabilities that are not current dependencies and may not share terminology with the project's mission.

4.4 Protocol interoperability does not establish authority

An agent can discover another agent. A model can call a tool. A service can deliver an event. None of those acts determines whether a project owner has authorized a consequential change.

The missing boundary is between possible cooperation and accepted project state.

4.5 Evidence remains downstream and inconsistent

Many integrations are adopted before their assumptions, costs, privacy effects, failure modes, and rollback behavior are packaged as one reviewable object. A generated patch without an evidence contract simply moves the burden from implementation to human investigation.

The proposed architecture therefore makes the Verification Packet a first-class product of integration, not an afterthought.

06

5. Definitions and system boundary

5.1 Project

A Project is a durably identified, independently governed undertaking with canonical state, artifacts, goals, constraints, authority, and revision history. It may be commercial, civic, scientific, personal, institutional, or artistic.

A Project is not identical to a website or repository. Those are projections and implementation surfaces.

5.2 Capability

A Capability is a bounded action, resource, model, dataset, service, method, standard, policy pathway, or coordination affordance that another Project could use.

5.3 Capability Beacon

A Capability Beacon is a signed or otherwise attributable machine-readable publication that announces a capability or material change to one.

5.4 Project Constitution

A Project Constitution is the machine-readable, human-authoritative boundary of purpose and permission. It records what the Project is trying to accomplish, what it must preserve, what it may spend, what data it may disclose, and which classes of change require which approval.

5.5 Scout

A Scout is a bounded process that observes public capability signals and compares them with the Project Constitution and current state. It may nominate opportunities. It does not own canon.

5.6 Fit Assessment

A Fit Assessment is a structured explanation of why a capability may or may not advance the Project, including goal alignment, conflicts, dependencies, uncertainty, cost, and likely value.

5.7 Adapter Forge

An Adapter Forge is a sandboxed builder that creates a candidate integration without mutating accepted project state.

5.8 Verification Packet

A Verification Packet is an evidence-bearing bundle containing exact inputs, implementation diff, test results, provenance, permissions, cost estimate, security analysis, limitations, rollback plan, and unresolved questions.

5.9 Acceptance

Acceptance is an authoritative act that promotes a verified candidate into recognized Project state. Verification may be automated. Acceptance remains assigned to a human or explicitly chartered institution.

5.10 Ambient integration

An Ambient Integration is a functional relationship discovered and proposed through the public capability environment without requiring a preexisting bilateral integration process.

07 6. Proposed architecture

The architecture has two public surfaces and one private authority plane.

6.1 Public capability surface

Capability providers publish Capability Beacons. A beacon may reference:

```
capability_id
provider_identity
version
human_summary
machine_description
interface_contracts
event_types
supported_tasks
intended_uses
prohibited_uses
security_requirements
data_practices
cost_model
evidenceandlimitations
compatibility
deprecation_policy
licenseandterms
contactanddispute_path
provenance
```

The beacon does not need to contain every field inline. It can provide stable links to OpenAPI descriptions, agent cards, model cards, terms, schemas, examples, attestations, and change history.

6.2 Project boundary surface

A Project exposes a selectively public Project Card and retains a more detailed private Constitution.

The public card may state:

```
project_id
mission
domain
current_capabilities
wanted_capabilities
public_constraints
accepted_protocols
proposal_endpoint
licensing posture
contact
```

The private Constitution includes sensitive constraints, budgets, data classifications, internal architecture, protected beneficiaries, risk thresholds, and approval rules.

This asymmetry is important. The Project must be understandable enough to receive useful proposals without publishing everything required to attack it.

6.3 Scout and opportunity graph

The Scout ingests beacons, standards updates, research releases, grants, policy changes, security advisories, service announcements, and project cards.

It compiles an Opportunity Graph:

```
capability -> project goal
capability -> missing requirement
capability -> current subsystem
capability -> conflicting constraint
capability -> required authority
capability -> expected evidence
capability -> cost and dependency
capability -> possible beneficiary
capability -> possible harm
```

A candidate is ranked not only by semantic similarity but by constitutional fit and evidence availability.

6.4 Candidate construction plane

A high-ranking opportunity enters the Adapter Forge with a scoped Work Order. The Forge receives only the minimum required project context and credentials. It builds in an isolated branch, fixture, simulation, or disposable environment.

The candidate cannot directly replace accepted state.

6.5 Verification and proposal plane

The system produces a Verification Packet and places it in a proposal inbox.

The operator should be able to answer:

- What newly became possible?
- Why does the scout think it matters here?
- What exact project state would change?

- What data and authority would cross the boundary?
- What did the candidate cost to build and what would it cost to operate?
- Which tests passed?
- What remains unknown?
- What is the rollback path?
- Who benefits?
- Who assumes new risk?
- What happens if the external provider changes or disappears?

6.6 Canon and receipt plane

If accepted, the Project records:

```
proposal_id
acceptedcandidatedigest
humanorinstitutional_authority
granted_capabilities
effective_time
evidence_digest
rollback_reference
externaldependencyidentity
terms_version
revieworexpiration_date
```

The accepted integration becomes a versioned relationship, not an invisible convenience.

08 7. Lifecycle of an ambient integration

A complete lifecycle contains ten phases.

1. Emit - a provider publishes a Capability Beacon.
2. Observe - project Scouts receive or discover it.
3. Match - the capability is compared with Project goals and constraints.
4. Explain - a Fit Assessment makes the proposed relevance inspectable.
5. Authorize exploration - a bounded Work Order permits candidate construction.
6. Forge - an adapter or configuration is built in isolation.
7. Verify - tests, security checks, costs, provenance, and rollback are packaged.
8. Accept or reject - the appropriate human or institution makes the consequential decision.
9. Commit and receipt - accepted state changes with exact lineage.
10. Reobserve - future provider changes, drift, cost, incidents, and alternatives trigger renewed review.

The lifecycle is recursive. An accepted integration can become a new capability that the Project itself publishes. Other projects may then discover and adapt around it.

This is how one release can propagate through a million domains without one organization centrally planning every use.

09 8. Worked example: Possum Party and Animal Keep

Assume the following accepted Possum Party goals:

- reduce animal injury during capture;
- reduce human injury during handling;
- produce auditable humane-release records;
- work in low-connectivity field conditions;
- minimize collection of personally identifying location data;
- keep per-trapper operating costs below a declared threshold.

A large platform publishes Animal Keep v1.2 with:

- on-device distress classification;
- an offline-first observation queue;
- a humane-handling event schema;
- a geofenced release verifier;
- a professional compliance export;
- a new paid telemetry service.

8.1 Scout finding

The Scout identifies four positive edges:

- distress classification -> reduce animal injury;
- offline queue -> low-connectivity operations;
- event schema -> auditable handling;
- compliance export -> professional reporting.

It also identifies two conflicts:

- telemetry service -> location privacy risk;
- subscription tier -> possible budget violation.

8.2 Fit Assessment

The Scout proposes a local-only integration path that uses the on-device classifier and offline event schema while refusing cloud telemetry. It marks the compliance export as unavailable under that path and identifies a licensing question.

8.3 Candidate

The Adapter Forge builds:

- a local classifier adapter;
- a schema translation layer;
- encrypted offline event storage;
- a redacted export;
- feature flags preserving the previous workflow.

8.4 Verification Packet

The packet includes:

- exact SDK version and hashes;
- device and model compatibility;
- fixture results across representative recordings;
- false-positive and false-negative summaries;
- battery and storage estimates;
- proof that telemetry endpoints were not contacted;
- license and cost analysis;
- rollback to the previous capture workflow;
- unresolved need for field validation with professionals.

8.5 Human decision

The human operator does not merely ask whether the code works. The operator decides:

- whether the distress model is reliable enough for advisory use;
- whether its errors could create false confidence;
- whether trappers understand that it does not replace judgment;
- whether the license permits the intended deployment;
- whether the privacy posture remains consistent with the Project;
- whether the benefit justifies the new dependency.

The integration may be accepted for a bounded pilot, rejected, or returned for stronger evidence.

Google and Possum Party never enter a formal partnership. Yet a public capability has become a governed local improvement.

10

9. From pairwise integration to a technological ecology

The most consequential property appears at scale.

In a conventional integration graph, edges are expensive. Each relationship requires discovery, negotiation, implementation, and maintenance. Large platforms prioritize edges with expected strategic return. Small projects form only the connections their people can see and afford.

In the Ambient Integration Web, the cost of proposing an edge falls. A capability can be emitted once and independently evaluated by many projects. A Project Constitution can guide many scouts and candidates without its human reexplaining the mission each time.

The graph begins to behave less like a directory of partnerships and more like an ecology.

A new camera model could trigger candidate accessibility upgrades, wildlife monitoring tools, manufacturing inspection systems, and educational instruments.

A new weather observation method could trigger adaptations in farms, transit systems, insurance models, neighborhood safety tools, and energy controllers.

A regulation could trigger proposed workflow and disclosure changes across every Project whose Constitution declares the affected jurisdiction and activity.

A public research method could be instantiated in domains its authors never anticipated.

The publisher does not need to know every downstream use. The downstream projects do not need to wait for an invitation. The joining occurs through machine-readable relevance, bounded construction, and local authority.

This is why the system is "disjointed, yet joined":

- ownership remains separate;
- identity remains separate;
- canon remains separate;
- authority remains separate;
- implementation may remain separate;
- functional opportunity becomes shared.

11

10. Economic and institutional consequences

10.1 Partnerships become one layer, not the only layer

Formal partnerships will remain important for guarantees, support, joint branding, data sharing, liability, and deep co-development. Ambient integration adds a lower layer of permissionless or generally licensed compatibility beneath them.

A Project can prove value before organizations negotiate a relationship.

10.2 Small projects gain a larger technical horizon

A small Project can have a narrow team and still maintain awareness across a large capability field. This does not remove resource inequality, but it may reduce the penalty for not having a dedicated partnerships, standards, research, and integration staff.

10.3 Capability publishers gain unplanned distribution

Organizations can publish capabilities with rich beacons and see adoption emerge in unexpected domains. Their incentive shifts from marketing every use case toward making capabilities inspectable, composable, safe, and attestable.

10.4 Intermediaries become constitutional matchmakers

New infrastructure providers may index beacons, host compatibility graphs, verify attestations, simulate costs, run adapter forges, insure integrations, or maintain revocation channels.

This creates risk of centralization. The index of possible relationships can become more powerful than the relationships themselves. Project cards and beacons should therefore remain portable, independently hostable, and discoverable through multiple indexes.

10.5 Software becomes institution-like

A serious Project stops being merely code plus a website. It develops:

- identity;
- charter;
- memory;
- permissions;
- incoming proposals;
- accepted relationships;
- audit history;
- dispute and exit mechanisms.

This resembles a small institution because autonomous adaptation requires governance, not only computation.

12

11. Security, misuse, and failure modes

Ambient opportunity is also ambient exposure. The architecture must fail closed.

11.1 Capability-beacon poisoning

An attacker publishes a beacon designed to match many project missions while hiding malicious behavior.

Controls include signed identity, reputation, reproducible artifacts, independent evidence, sandboxing, network observation, and distrust of self-authored claims.

11.2 Constitution extraction

A Scout may reveal sensitive project strategy by querying external services or publishing overly detailed wants.

Controls include local matching, selective disclosure, private indexes, query blinding, compartmentalized scouts, and public/private Constitution separation.

11.3 Proposal flooding

Projects may receive enormous numbers of low-value or promotional proposals.

Controls include constitutional filters, sender cost, rate limits, reputation, proof-of-work or proof-of-relevance, local ranking, and an explicit no-solicitation policy.

11.4 Dependency capture

A useful integration can gradually make a Project operationally dependent on one provider.

The Verification Packet should include exit cost, local fallback, data portability, price sensitivity, substitute capabilities, and review dates.

11.5 Silent scope expansion

An adapter accepted for one narrow use may later request more data, actions, or budget.

Capabilities must be leased by scope and version. New permissions require a new acceptance event.

11.6 Evidence theater

A generated report can look rigorous while testing irrelevant conditions or omitting failures.

Evidence contracts must be declared before evaluation where possible. Raw artifacts, exact configurations, negative results, and limitations remain inspectable. Model output cannot self-award PASS.

11.7 Cross-project contamination

A scout serving multiple Projects may leak context, code, or priorities between them.

Project state, credentials, caches, and candidate artifacts require explicit isolation. Semantic similarity does not authorize reuse.

11.8 Automated monoculture

If many Projects adopt the same highly ranked capability, one failure or policy change can propagate widely.

Diversity, substitute paths, staggered adoption, bounded pilots, and dependency concentration metrics become system-level safety tools.

11.9 Human abdication

The most serious failure is cultural: people may treat a well-presented candidate as a decision already made.

The interface must preserve the difference among nomination, verification, authorization, and acceptance.

13

12. Relationship to Glyphd architecture

The Ambient Integration Web composes several existing Glyphd Labs boundaries.

12.1 Actor before agent

The Project is the persistent Actor. Scouts and Adapter Forges are temporary Agent processes. They receive bounded context, propose patches, and terminate. The Project persists when no model runs.

12.2 Project Core and canonical state

The Project Constitution, accepted integrations, authority, and receipts belong to durable canonical state. Search indexes, compatibility maps, and recommendation surfaces are projections. They cannot become a hidden second authority.

12.3 CAS-native coordination

Capability Beacons, Work Orders, candidate adapters, Verification Packets, and receipts should be content-addressed or otherwise exactly identified. A later reviewer must know which inputs and terms produced which proposal.

12.4 Glyph Tokens

Mission, constraint, authority, uncertainty, evidence, and action can be represented as typed semantic carriers rather than flattened prose. Structure does not create trust; source and authority do.

12.5 Human in master control

Discovery can be broad. Consequential authority remains visible, scoped, revocable, and receipted. No external capability, scout, or agent can silently award itself a new role.

The compact invariant is:

> *Scouts nominate. Forges construct. Verifiers test. Humans or chartered institutions accept. Canon records.*

14

13. Implementation path

A credible implementation should begin narrowly.

Phase 1: local project scout

One Project maintains a private Constitution and watches a curated set of feeds, release notes, package ecosystems, and standards sources. The Scout produces Fit Assessments only.

Success means relevant nominations with low noise. No code is generated.

Phase 2: bounded candidate forge

The Project permits candidate branches for a limited class of integrations, such as developer tooling or non-sensitive public-data APIs. Every candidate includes tests, cost, permissions, and rollback.

Success means truthful, reviewable proposals rather than autonomous adoption.

Phase 3: public Capability Beacons

A small set of providers publishes a common beacon profile referencing existing OpenAPI, MCP, A2A, model-card, license, and provenance artifacts.

Success means multiple scouts can parse the same capability publication without custom scraping.

Phase 4: Project Cards and proposal endpoints

Projects publish selectively safe descriptions and receive structured proposals.

Success means an independently developed scout can nominate a relevant capability to a Project without prior bilateral configuration.

Phase 5: multi-project compatibility graph

Public or federated indexes connect capabilities, goals, constraints, evidence, and known integrations.

Success means useful discovery without one registry becoming mandatory or authoritative.

Phase 6: institutional governance

Organizations define delegated approval, budget, audit, review, revocation, and dispute policies. High-consequence domains require stronger evidence and named accountable authorities.

Success means automation scales without dissolving responsibility.

15

14. Experimental method and falsification

The central future claim is not established by existing standards. It requires experiments.

14.1 Opportunity-discovery benchmark

Construct a corpus of Projects with explicit goals and constraints plus a time-ordered stream of capability releases.

Compare:

- keyword alerts;
- ordinary semantic search;
- dependency-only bots;
- Constitution-aware Scouts;
- human expert discovery.

Measure:

- relevant opportunity recall;
- false proposal rate;
- time to first useful proposal;
- constitutional conflict detection;
- explanation quality;
- operator review time;
- diversity of discovered capabilities.

14.2 Candidate-quality benchmark

For accepted opportunity pairs, compare human-built integrations with Scout-plus-Forge candidates.

Measure:

- functional test pass rate;

- unrequested scope;
- security findings;
- permission minimization;
- cost accuracy;
- rollback success;
- stale evidence;
- reviewer disagreement;
- maintenance burden after provider updates.

14.3 Human-authority study

Compare interfaces that present:

1. a confident recommendation;
2. a code diff;
3. a full Fit Assessment and Verification Packet;
4. a proposal with explicit Constitution conflicts and authority scope.

Measure whether operators correctly distinguish possible, built, verified, authorized, and accepted states.

14.4 Network simulation

Model thousands of Projects and capability releases with varied goals, trust, budgets, and dependencies.

Measure:

- useful edge formation;
- propagation speed;
- concentration risk;
- cascading failure;
- duplicate effort;
- exploit spread;
- recovery under revocation;
- index centralization.

14.5 Kill criteria

The Ambient Integration Web should be rejected or substantially narrowed if:

- Constitution-aware matching does not materially outperform competent search and curated feeds;
- proposal noise exceeds operator capacity;
- candidate verification cost approaches manual integration cost;
- sensitive Project intent cannot be protected;
- authority boundaries are routinely misunderstood;
- dependency concentration grows faster than useful interoperability;

- malicious beacons or evidence theater remain difficult to detect;
- human operators approve harmful changes at higher rates because automation creates false legitimacy.

16

15. Current evidence state and limitations

The paper’s historical claim is source-reviewed: standards and deployed systems already implement change feeds, publish-subscribe, interface description, automated pull-request updates, desired-state reconciliation, federation, agent interoperability, scoped authorization, and provenance.

The proposed composition is not implemented as one accepted system. No qualifying result demonstrates that mission-aware Scouts can discover useful cross-organizational integrations at acceptable precision, cost, privacy, and risk.

Important limitations include:

- Project purpose is difficult to formalize without losing nuance.
- Human goals conflict and change.
- Public descriptions can expose strategy and attack surface.
- Many capabilities cannot be responsibly tested in a sandbox.
- Legal terms may prohibit automated adaptation or downstream use.
- Model-generated adapters may create subtle security and maintenance debt.
- Verification cannot fully predict social consequences.
- Organizations may manipulate indexes and compatibility scores.
- Small Projects may lack the capacity to review even excellent proposals.
- Human acceptance can become ceremonial rather than substantive.
- A global compatibility layer could centralize power despite decentralized protocols.
- The word "project" covers entities with radically different authority and accountability structures.

The proposal is therefore a research program, not a claim that the autonomous web is ready to deploy.

17

16. Reproduction and public artifacts

This publication package contains:

- the authored paper body;
- a claim ledger with bounded evidence states;
- a source manifest using primary standards and official documentation;
- four conceptual architecture plates;
- revision history;
- a public-disclosure boundary;
- BibTeX and JSON metadata;

- a downloadable PDF.

No executable Ambient Integration runtime or benchmark result is included. The diagrams are conceptual. The worked example is hypothetical.

18

17. The human operator: author of the should

The easiest mistake is to imagine that the future human operator becomes unnecessary because software can find, build, test, and maintain connections.

The opposite is more accurate.

As execution becomes cheaper, the number of possible actions increases. The scarce resource becomes not the ability to make a change but the ability to decide which changes deserve to become real.

The human operator’s role moves upward from performing every integration to governing the field of possible integrations.

17.1 The operator authors purpose

A Scout cannot infer a legitimate mission merely from behavior, usage analytics, or revenue. Those signals describe what has happened, not what ought to happen.

The operator states:

- whom the Project serves;
- which outcomes matter;
- which values may not be traded away;
- which needs are real;
- which forms of growth are unwanted;
- which uncertainties must remain visible.

This is constitutional work.

17.2 The operator defines the boundary of acceptable autonomy

The operator decides what the system may:

- observe;
- spend;
- disclose;
- download;
- execute;
- modify;
- publish;
- commit;
- renew;
- revoke.

Low-risk, reversible maintenance may receive delegated authority. High-consequence changes may require named approval, external review, or collective governance.

The operator does not need to click every harmless action. The operator must remain the source of the authority under which those actions occur.

17.3 The operator judges consequences that cannot be reduced to tests

A candidate can pass functional checks and still be wrong.

It may weaken privacy, deskill a profession, shift risk to vulnerable people, create dependency on a hostile provider, violate a community norm, or optimize the measurable part of a mission while damaging its meaning.

The human operator asks not only, "Does it work?" but:

- What behavior will this encourage?
- Who gains power?
- Who becomes dependent?
- Who bears failure?
- What becomes harder to undo?
- Which beneficiary is absent from the data?
- What will the Project become if it accepts many changes like this?

17.4 The operator handles exceptions and conflict

Constitutions cannot anticipate every situation. Goals collide. Evidence remains incomplete. A promising capability may violate a rule written for a different context.

The operator interprets, amends, or refuses. This is not a flaw to automate away. It is the place where responsibility is exercised.

17.5 The operator accepts and remains accountable

Verification answers whether evidence satisfies a declared test. Acceptance answers whether the Project will own the consequence.

A model cannot self-award acceptance because it cannot inherit the moral, legal, social, or institutional accountability of the operator. An automated institution may receive delegated authority, but that delegation itself must be authored, inspectable, contestable, and revocable.

17.6 The operator preserves plurality

A billion-dimensional integration web could converge rapidly on whatever capabilities are easiest to discover, cheapest to adopt, or best promoted by dominant indexes.

The human operator can preserve local difference:

- choosing a smaller provider;
- retaining a manual practice;
- refusing surveillance;

- preferring resilience over optimization;
- keeping data local;
- protecting a cultural or professional norm;
- declining growth;
- maintaining more than one path.

Plurality is not inefficiency by default. It is a defense against monoculture and a source of future invention.

17.7 The operator becomes a gardener of becoming

The mature operator does not micromanage every machine action, nor surrender the Project to an optimization loop. The operator tends a living boundary between purpose and possibility.

The system continually asks:

> *What has appeared in the world that could help this Project become more fully what it intends to be?*

The human answers:

> *Is that becoming still ours?*

That is the final division of labor.

The machine expands the field of visible possibility. It searches, compares, constructs, tests, monitors, and remembers. The human authors purpose, grants authority, interprets conflict, accepts consequence, protects plurality, and can always say no.

The future web may be disjointed in ownership and joined in function. Its human operators remain the authors of the should that gives those connections direction.

19 Architecture illustration briefs

- P29-F01 - The existing fragments: feeds, publish-subscribe, interface descriptions, dependency bots, controllers, federation, agent protocols, authorization, and provenance shown as mature but disconnected layers.
- P29-F02 - The ambient integration loop: Capability Beacon -> Scout -> Fit Assessment -> bounded Work Order -> Adapter Forge -> Verification Packet -> human acceptance -> canonical receipt -> renewed observation.
- P29-F03 - Disjointed, yet joined: independently owned Projects connected by unofficial, evidence-bearing functional edges while retaining separate canon and authority.
- P29-F04 - The human operator’s authority stack: machine discovery and construction below; human-authored purpose, boundaries, exception judgment, acceptance, accountability, and revocation above.

20 References

1. Nottingham, M., and Sayre, R. RFC 4287: The Atom Syndication Format. IETF, 2005. https://www.rfc-editor.org/rfc/rfc4287.html

2. W3C. WebSub. Recommendation, 2 June 2026. https://www.w3.org/TR/websub/

3. Cloud Native Computing Foundation. CloudEvents Specification. https://github.com/cloudevents/spec

4. OpenAPI Initiative. OpenAPI Specification v3.2.0. 19 September 2025. https://spec.openapis.org/oas/v3.2.0.html

5. W3C. Web of Things (WoT) Thing Description 1.1. Recommendation, 5 December 2023. https://www.w3.org/TR/wot-thing-description11/

6. GitHub. Dependabot version updates. https://docs.github.com/en/code-security/concepts/supply-chain-security/dependabot-version-updates

7. Kubernetes. Controllers. https://kubernetes.io/docs/concepts/architecture/controller/

8. Kubernetes. Operator pattern. https://kubernetes.io/docs/concepts/extend-kubernetes/operator/

9. W3C. ActivityPub. Recommendation, 23 January 2018. https://www.w3.org/TR/activitypub/

10. Model Context Protocol. Specification 2025-11-25. https://modelcontextprotocol.io/specification/2025-11-25

11. A2A Protocol. Agent2Agent Protocol Specification, latest released version 1.0.0. https://a2a-protocol.org/dev/specification/

12. Hardt, D. RFC 6749: The OAuth 2.0 Authorization Framework. IETF, 2012. https://www.rfc-editor.org/rfc/rfc6749.html

13. in-toto. Getting started and specification overview. https://in-toto.io/docs/getting-started/

14. Glyphd Labs. Actor Before Agent. P06, 2026.

15. Glyphd Labs. Agents That Work on Addressed State. P09, 2026.

16. Glyphd Labs. MCE-1: Human in Master Control. P21, 2026.

17. Glyphd Labs. Chat - Plan - Make. P22, 2026.

18. Glyphd Labs. Glyph Tokens. P26, 2026.

21 Revision history

- 0.1.0 - 2026-07-30: Initial public-safe working paper. Defines the Ambient Integration Web, traces its existing technical fragments, specifies the project-level integration lifecycle, and states the human operator’s role.

22 Public-disclosure and IP boundary

This paper publishes a conceptual architecture, terminology, protocol roles, safety boundaries, evaluation plan, and historical synthesis. It does not disclose private credentials, private project constitutions, unpublished implementation code, restricted matching heuristics, proprietary ranking thresholds, private source materials, or claim-ready legal analysis. The hypothetical Possum Party example is illustrative and does not represent a deployed product or an affiliation with any named platform.