Layered Agentic Retrieval:
A Publishing Architecture for Agent-Native Web Surfaces

Author: Francesco Marinoni Moretto
ORCID: 0009-0004-0096-4712
Contact: francesco.marinoni.moretto@gmail.com
Status: Working paper
Date: May 2026

Abstract

As AI agents transition from read-only assistants to autonomous actors, the human-readable web is failing them. Current retrofitting approaches, such as schema.org-enhanced HTML or Retrieval-Augmented Generation (RAG), force agents to parse unstructured or weakly-structured corpora or expend finite token budgets on human-centric rendering pipelines. In response, a parallel publishing layer is taking shape: major AI operators — including OpenAI, Anthropic, Stripe, and Google — are increasingly exposing machine-legible surfaces optimized for autonomous traversal, deterministic retrieval, and reduced token consumption. This paper identifies and analyzes this emerging publishing pattern as Layered Agentic Retrieval (LAR).

LAR is a publisher-side architectural commitment to surface deterministic, machine-legible data designed explicitly for agent traversal. We articulate LAR's three core commitments — plain-text substrates, progressive disclosure, and workflow-organized hierarchies — and situate it within the lineages of RESTful hypermedia and the Semantic Web. By delineating the limits of current retrofitting approaches, this paper establishes LAR as a dedicated institutional layer through which publishers can expose machine-legible retrieval, policy, and operational surfaces to autonomous agents — the structural artifact onto which cryptographic attestation attaches and on which emerging legal provenance frameworks can operate.

Keywords: agentic commerce, agent-mediated retrieval, web architecture, knowledge graph, semantic web, RESTful hypermedia

Abbreviations

ACP	Agentic Commerce Protocol (OpenAI/Stripe).
ADK	Agent Development Kit (Google).
A2A	Agent2Agent (Linux Foundation, originally Google).
HATEOAS	Hypermedia as the Engine of Application State (Fielding, 2000).
HAL	Hypertext Application Language.
JSON-LD	JSON for Linked Data (W3C).
LAR	Layered Agentic Retrieval (this paper).
MCP	Model Context Protocol (Anthropic, server-side).
RAG	Retrieval-Augmented Generation.
RDF	Resource Description Framework (W3C).
SPT	Shared Payment Tokens (Stripe, used by ACP).
UCP	Universal Commerce Protocol (Google + retail coalition).
WebMCP	Browser-native tool exposure standard (W3C Web ML CG, distinct from MCP).

1. Introduction

A small set of operators — OpenAI, Anthropic, Stripe, Apideck, Google's Agent Development Kit team, WordLift, Microsoft's NLWeb group — are publishing knowledge to AI agents in a shape that does not look like a website. The pieces of this shape have local working names; the whole does not. This paper proposes a name — Layered Agentic Retrieval (LAR) — and articulates the architectural commitments that distinguish what these operators ship from conventional web publishing or crawler-friendly retrofit.

Each of these operators publishes a separate agent-facing layer, in order to expose content optimized for agent traversal rather than human rendering. While this pattern is already emerging in production, the vocabulary that lets merchants, researchers, and policy actors discuss it as a single architectural object is missing. The paper's contribution is naming and framing, not formal specification or benchmarking. It situates LAR in its theoretical lineage, distinguishes it from adjacent approaches (notably Retrieval-Augmented Generation and schema.org-enhanced HTML), and identifies the empirical and legal questions that remain open.

2. The pattern emerging in production

OpenAI's developer documentation site exposes `/codex/llms-full.txt`: a single markdown file containing the complete documentation, regenerated on every doc update.[1] Google's Agent Development Kit ships an equivalent.[2] Apideck publishes a three-tier disclosure pattern: `/llms.txt` as a summary index, `/llms-full.txt` as an enumerated index of every doc page, individual `.md` files at each leaf, and a programmatic `md/index.json` for machine traversal.[3] Other developer-tool vendors — Meteor, FastMCP, assistant-ui, Chkk, Mintlify — ship the same shape.[4] Anthropic's published skills are organised similarly: each skill exposes its name and description in YAML frontmatter (the manifest), with SKILL.md as the entry point and additional resources — markdown documentation, executable scripts, and bundled assets — in subdirectories the agent loads only when its task requires them.[5]

The surfaces[6] examined here share observable structural properties: the markdown and JSON files appear to be regenerated alongside doc updates rather than scraped from rendered HTML. This pattern is consistent with shared upstream source material with the human-facing artifacts, though it does not prove it.

These concrete forms differ in their surface organisation, but three properties recur across all of them:

Root declaration. A root surface (whether a single file or a manifest) declares what is published and links to deeper material.

Format separation. Narrative content is held in markdown while structured content requiring deterministic parsing is held in JSON, with the two kinds separated rather than interleaved.

Explicit machine-resolvable links. Links between files are explicit and machine-resolvable, allowing an agent to fetch the root, decide which subtree its task requires, and walk down only as far as it needs — without rendering, scraping, or inferring structure from HTML.

The protocol layer is taking shape in parallel. Microsoft's NLWeb defines how a website exposes structured data to agents through a knowledge-graph-backed natural language query interface.[7] Google's Universal Commerce Protocol (UCP), launched at NRF 2026, standardises commerce capabilities across AI surfaces.[8] OpenAI and Stripe's Agentic Commerce Protocol (ACP), open-sourced in September 2025, addresses the same transactional layer through agent-mediated execution within conversational surfaces.[9] WebMCP, a W3C Community Group Report (latest draft 10 February 2026), exposed in early 2026 Chrome 146 preview builds behind the "Experimental Web Platform Features" flag, lets a website declare itself as a tool surface for in-browser agents.[10]

These initiatives address different layers — site-level discovery, commerce capability negotiation, agent-mediated transaction, browser-native tool exposure — but share an underlying assumption: agents need surfaces designed for them, and those surfaces will be declared, not inferred.

External infrastructure providers are operationalising this layer distinction. Google's Search Central guidance, published 15 May 2026, explicitly states that publishers "don't need to create new machine readable files, AI text files, markup, or Markdown to appear in generative AI search," while a parallel section recognising autonomous agents notes that "protocols like Universal Commerce Protocol (UCP) are emerging that will allow Search agents to do more."[11] Cloudflare's Agent Readiness Score, launched 17 April 2026, operationalises assessment of publisher-side agent infrastructure across four dimensions (discoverability, content, bot access control, capabilities) plus commerce protocols, with adoption tracked across 108,595 successfully scanned domains as of May 2026.[12] The asymmetry is informative: Google scopes its guidance to Search optimisation while signalling that agentic experiences require different infrastructure; Cloudflare operationalises that infrastructure with empirical adoption data. The publisher/agent distinction LAR articulates is becoming externally recognised in the ecosystem the paper describes.

3. Three architectural commitments

The operators building these surfaces deliberately, rather than as accidental byproducts of documentation tooling, share three commitments that distinguish their output from retrofit.

Plain text instead of pages. Markdown for narrative, JSON for facts, YAML for configuration. No DOM, no JavaScript, no rendering pipeline. An agent parses these formats deterministically in milliseconds, with no model call required for extraction. The llms.txt specification states the rationale directly: context windows are too small to handle most websites in their entirety, and converting complex HTML pages (with navigation, ads, and JavaScript) into LLM-friendly plain text is difficult and imprecise.[4]

Progressive disclosure instead of flat exposure. A manifest at the root, topic indexes one layer down, detail files at the leaves. The agent walks the tree to the depth its task requires and stops. Token budget is controlled by the agent, not by the publisher's page boundaries. This is the structural inverse of a sitemap. Anthropic's Claude Skill Design has codified this pattern as progressive knowledge disclosure for tool packaging, where agents see only skill names and short descriptions initially and load detailed files only when needed.[5] MajesticLabs's hierarchical-agents pattern applies the same principle to codebase documentation.[13]

Organised by workflow instead of by topic. A merchant's human site is organised by what the merchant wants to say: about us, our story, our products, contact. A LAR surface is organised by what the agent was tasked to do, and the discretion the principal delegated for doing it: availability, inventory, policies, pricing, the endpoint that commits the transaction. The structure mirrors the agent's mandate, not the merchant's marketing.

Each commitment is explicitly stated in the documentation of the operators cited above; what is proposed here is the framing that holds the three together as a single architectural pattern.

The theoretical justification is the economics of the LLM context window: an agent operating under finite token budget pays a cost proportional to every byte loaded, and topic-organised structures force the agent to traverse material organised around the publisher's narrative concerns before reaching the data its task actually requires.

Workflow organisation collapses this asymmetry by aligning the structure of the surface with the structure of agent decisions, allowing token usage to scale with task complexity rather than with publisher verbosity.[14]

The discipline of building a good LAR surface is therefore not to design the path the agent will follow, but to remove obstacles from every path the agent might take. The work is to expose relevant data efficiently, without forcing the agent through structure that reflects the publisher's mental model rather than the agent's task.

These three commitments are what make the architecture layered rather than retrofit, agentic rather than crawler-friendly, retrieval-shaped rather than render-shaped.

LAR is not Retrieval-Augmented Generation. RAG is a consumer-side pattern that retrieves probabilistically over unstructured or weakly-structured corpora to augment generation; LAR is a publisher-side commitment to structure the source so that retrieval can be deterministic when the consuming agent respects the layered structure.[15]

Two terms are used in this paper with deliberate distinction, doing different work even when they describe the same artifact.

Machine-readable describes formats a machine can deterministically parse — the standard sense established by RDF, HATEOAS, and the broader Semantic Web tradition.

Machine-legible describes content a machine can meaningfully act upon. JSON exposing five generic product attributes is machine-readable; JSON exposing twenty content-rich attributes that capture provenance, occasion-fit, and operational constraints is machine-legible.

The conceptual distinction between format and meaning has surfaced in adjacent literature on digital product labels and Agent Experience design; the two-term vocabulary proposed here is intended to make the distinction usable in operator and policy discourse.

LAR commits to both: machine-readable formats (plain text, JSON) carrying machine-legible content (workflow-organised, identity-bound, attestation-ready).

4. Theoretical lineage and recent convergence

Although the vocabulary proposed here is new, the pattern it names is not. Manifest-driven traversal of linked, machine-readable resources has scholarly lineage.

The hypermedia-driven REST architectures articulated by Fielding (2000) — commonly known as HATEOAS — describe this shape: a client fetches a root resource, discovers links at runtime, and traverses progressively to fulfil its task.[16] The early Semantic Web vision of Berners-Lee, Hendler and Lassila (2001) extended the pattern to machine-readable linked data, anticipating agents that would compose actions across distributed semantic resources.[17]

LAR is best understood as an instantiation of this lineage adapted to the LLM-agent consumer: plain text and JSON calibrated for context-window absorption rather than HAL or RDF, organisation by task-workflow rather than resource-entity, consumer that performs semantic parsing on prose rather than following typed links. The architectural innovation, where one exists, lies in the calibration to the consumer rather than in the traversal pattern itself.

Recent practitioner thinking has converged on a parallel architectural pattern from the consumer side. Andrej Karpathy (OpenAI founding member, former Tesla AI Director) published his LLM Wiki sketch (2026) via a post on X and a gist on GitHub — the genre by which influential AI thinking now first reaches the field — describing an LLM agent ingesting a user's raw sources, compiling them into a persistent markdown knowledge base, and maintaining it across queries.[18]

The sketch arrives at architectural decisions that overlap with LAR's publisher-side commitments: markdown substrate, three-layer separation, rejection of RAG as default. Karpathy's wiki is what an agent maintains for one user's consumption; LAR is what a publisher exposes for any agent's traversal. The convergent design choices on opposite ends of agent-mediated information flow are the more interesting observation than the architectural overlap itself.

A strict architectural critique might observe that the systems discussed above operate at different layers of the software stack: llms.txt addresses documentation discovery, MCP standardizes tool invocation, and ACP governs transactional execution. LAR does not claim these protocols are equivalent, nor that they require a shared implementation layer.

The claim is narrower and structural: finite-context autonomous agents create common economic pressures across discovery, retrieval, and execution. In response, operators are increasingly exposing machine-legible institutional surfaces designed to reduce ambiguity, traversal cost, and rendering overhead for autonomous systems. LAR is proposed as a publisher-side conceptual vocabulary for understanding this convergence.

5. The limits of retrofit

The dominant publisher-side approach to making content readable by agents is structured data embedded in HTML using schema.org vocabulary, typically expressed as JSON-LD inside `<script>` tags. This is the consensus enterprise approach today: a website remains organised for human consumption, with machine-readable annotations layered on top. Search engines, retrieval systems, and most LLM-mediated agents can extract the structured data without the merchant restructuring anything.

Any proposal for a distinct publisher-side architecture has to engage with one question: why is schema.org-via-JSON-LD not enough?

A controlled experiment by Volpini and colleagues (2026) measured retrieval accuracy across four domains — editorial, legal, travel, and e-commerce — comparing the same structured data embedded as JSON-LD inside HTML script tags against materialisation as agent-navigable entity pages. The materialised version improved retrieval accuracy by 29.6% under the experimental conditions disclosed.[19] On the specific question of embedded structured data versus materialised entity pages for agent retrieval, this is the only direct empirical measurement we are aware of.

Adjacent evidence from tool-mediated retrieval benchmarks likewise reports substantial performance gains for structured access over open web retrieval, including a 71 percentage point gap in financial data retrieval tasks for a frontier LLM agent (Kim & Huang, 2026).[20] The WordLift team frames this from the agent's perspective as *Agent-Orchestrated Retrieval*: how agents traverse linked data and aggregate context across entity boundaries.

LAR and enhanced entity pages are best understood as projections of the same publisher-side resource: a canonical knowledge graph describing what an institution offers, identifies, and authorises.[21]

Entity pages project this graph by entity, optimising for retrieval-by-identification — they answer the question *what is this entity in machine-readable form?* LAR projects the same graph by workflow, optimising for retrieval-by-execution — it answers *what must an agent do, and in what order, to complete a task?*

The two projections share infrastructure (vocabulary, identifier schemes, attestation primitives) and address different consumer behaviours. They are complementary, not competitive: a publisher with a robust canonical graph can produce both, and a sophisticated agent may consume both within the same task.

The crucial implication is that LAR is not a substitute for canonical structuring. A merchant who builds a LAR surface without an underlying canonical graph publishes leaf files that may be plain text and well-organised but reference inconsistent entities, expose workflows that misalign with how products are catalogued, and produce contradictions between the LAR surface and the rest of the institution's data.

6. A LAR surface in concrete form

To make the architecture concrete, consider Pelletteria Vanelli, a composite reference architecture designed to illustrate a mid-market Florentine leather goods merchant — third-generation, ~80 SKUs, premium artisanal positioning, established e-commerce site, in-house digital team.[22] The human-facing site is organised the way most merchant sites are organised: a homepage with brand storytelling, a product catalog browsable by category, an about page, a blog. An agent visiting this site has to render JavaScript, parse HTML, infer structure from CSS classes, and reconstruct the catalog from page-level templates — a sequence on which agent performance against published benchmarks remains low.[23]

A LAR surface for the same business exposes a different shape. The merchant publishes a structured manifest at `/.well-known/lar.json`, declares its presence in `robots.txt` alongside the existing `Sitemap:` directive, and references it from the HTML head of rendered pages. The

manifest itself is short and declarative; deeper material is held in linked files within the publisher's own namespace.

A schematic example, with the discovery surface that points agents to the manifest:

```
# /robots.txt
User-agent: *
Allow: /
Sitemap: https://example.com/sitemap.xml
LAR: https://example.com/.well-known/lar.json
```

The manifest referenced by that directive resolves to the following tree (full JSON schema in Appendix A.1):

```
/.well-known/lar.json   (root manifest, JSON)
  → /lar/about.md               (brand context, narrative)
  → /lar/catalog/index.md       (product catalog)
  → /lar/availability.json      (real-time stock)
  → /lar/policies/index.md      (returns, shipping, warranty)
  → /lar/pricing.json           (current pricing by SKU)
  → /lar/checkout/spec.json     (transaction endpoint)
```

The root manifest is JSON because it carries content requiring deterministic parsing — entity identification, operations declarations with protocol bindings, attestation primitives, schema version. Narrative content (institutional identity, brand context, occasion-fit prose) lives in linked markdown files where prose is the natural format and LLM consumption is direct. Volatile state (availability, pricing) lives in JSON or behind API endpoints. The format-follows-purpose discipline articulated in §3 applies to the manifest itself: each layer of the surface uses the format best suited to its consumer. (The Identity template for the about.md layer is provided in Appendix A.2.)

The manifest is the first thing an agent reads. It is short, declarative, and explicit. What the agent does next depends on the task. A research task descends into `/catalog/`. A purchase task descends into `/availability`, then `/pricing`, then `/checkout`. A complaint task descends into `/policies/`. The surface does not prescribe these paths; it enables them. Each leaf is a structured unit that links explicitly to related leaves the agent may need. The narrative ("artisanal Florentine craftsmanship since 1962") is preserved in `/lar/about.md` for agents whose tasks require brand context. It is not in the way of agents whose tasks do not.

The location at `/.well-known/lar.json` follows RFC 8615 well-known URI conventions; the filename is offered as a working proposal anchored to the framework name. Refinement of the canonical filename is open to community discussion through the project repository and discussion channel referenced at the conclusion of this paper. Adopters and reviewers are invited to participate in convention finalization. Should community consensus converge on an alternative within the well-known namespace, the architectural pattern documented here applies unchanged: only the canonical filename would be revised in subsequent versions.

LAR is not a stripping exercise. The architecture does not require removing storytelling, brand voice, or intangibles. A surface hollowed out to its measurable attributes would serve only the narrowest class of tasks. For every other task — selection, comparison, contextual evaluation — the intangibles are part of what the agent must reason about. The discipline is structuring richness so that the agent can choose how much of it to load, not subtracting it.

7. Architecture at scale

The reference architecture above is the smallest case: ~80 SKUs, weekly catalog refresh, policies stable for a year. A retailer with fifty thousand SKUs and pricing that changes every fifteen minutes cannot publish static markdown for everything. An editorial publisher with a paywall that moves article-by-article cannot freeze its current state into files. The architecture has to scale across these cases without becoming three different architectures.

It does so through a separation of concerns into three layers:

Identity is what the institution is and what it stands for: legal entity, mission, certifications, taxonomies, and the policies (return, shipping, privacy, citation, access) that govern operations. This layer is stable and lives well in markdown and structured static files.

Current state is what the institution offers right now: inventory, availability, pricing, schedule, paywall status, capacity. Its medium follows its volatility — static JSON when data refreshes daily or weekly, API endpoints when it refreshes by the hour or minute.

Operations is what an agent can do via the surface: purchase, subscribe, cite, reserve, donate, apply, access. This layer declares the operations the institution authorises and binds each operation to an endpoint capable of executing it under a stated protocol — UCP, ACP, MCP, or proprietary. The protocol binding is broader than conventional APIs alone. Depending on the agent ecosystem a LAR surface is designed to serve, declared operations may reference protocol-bound transaction surfaces (ACP, UCP), MCP-compatible tool resources, browser-native tool declarations (WebMCP), skill bundles, or other ecosystem-specific machine-actionable descriptors exposed under publisher control.

The three layers are not three files. They are three concerns. A merchant in the simplest case publishes them as three branches of a single static tree. A merchant at scale publishes Identity as static markdown, Current State as a small set of named API endpoints, and Operations as a declarative section in `lar.json` enumerating endpoints with their protocol bindings. The same root surface, two operationally different shapes.

Progressive disclosure applies to endpoints just as it applies to files: the surface does not enumerate every individual product or price point at the root. It declares the layer, declares a small set of canonical access paths, and lets the agent unfold downward only as far as its task requires. A retailer's current-state does not list fifty thousand SKUs in `lar.json`; it declares an availability endpoint and a pricing endpoint, each serving agent queries for specific SKUs at specific moments, with declared freshness windows and declared governing policies. The fan-out happens at the endpoint, not in the manifest.

This dynamic capability invites a critical architectural question: if LAR relies on API endpoints for volatile state, how does it differ from a standard REST API documented via OpenAPI?

The distinction lies in the institutional envelope. An OpenAPI specification defines a *technical contract*: how does this endpoint work, and what is its schema? It is optimized for human developers integrating systems ahead of time.

A LAR surface, conversely, defines an *institutional declaration*. An endpoint declared within a LAR manifest (`lar.json`) can be cryptographically bound to the institution's identity and policies when the manifest includes attestation primitives (Appendix A.1). When an agent

queries a dynamic LAR endpoint for pricing, it does not just receive a JSON response; it receives a response governed by the `policies/index.md` file linked at the root, signed under the same institutional attestation where the manifest declares one. OpenAPI tells the agent *how* to execute a transaction; LAR tells the agent *who* is legally authorizing the transaction and under *what terms*. Without this declarative upper layer, agents are forced to rely on bespoke integrations or probabilistically scrape human-facing interfaces — approaches that fail to scale across the long tail of internet merchants.

For merchants already publishing OpenAPI specifications, LAR sits above them as the institutional layer: existing API contracts continue to govern technical execution, while the manifest declares institutional authorisation and policy binding.

8. Open questions

LAR is an emerging pattern, not a settled standard. Several questions are unresolved.

Empirical questions.

Empirical task-success benchmarks. Volpini and colleagues measured retrieval accuracy on entity pages.[19] Existing agent web-navigation benchmarks — WebArena (Zhou et al., 2023) and Mind2Web (Deng et al., 2023) — provide methodological precedent for the agent task harness; what they do not provide is the publishing-side comparison: retrofit HTML, schema.org/JSON-LD entity pages, and workflow-organised LAR evaluated across shared agent tasks.[23] While this paper provides the foundational architectural framing, empirical validation against retrofit baselines remains open. A planned field study (Marinoni Moretto & Antichi, forthcoming 2026), conducted in partnership with digital strategy firm Advertaria, will measure token-efficiency and agent task-success on a cohort of mid-market Italian e-commerce merchants.[24] Benchmark results will be published in subsequent revisions of this working paper.

Falsifiability. The claim that LAR represents an emerging architectural pattern is testable. It would be disconfirmed by operator divergence on basic shape (manifest plus linked files plus workflow-organisation), retraction rather than expansion of publishing surfaces over time, or empirical evidence that retrofit retrieval consistently outperforms structured publication for agent task completion.

Coordination questions.

Manifest standardisation. Several discovery formats are circulating in parallel: llms.txt as a documentation-style index, MCP server manifests for backend tool exposure, A2A agent cards (now under Linux Foundation governance), NLWeb's site-level conversational interface, WebMCP's browser-native tool declaration, ACP's agent-mediated commerce flow, UCP's commerce-specific manifest, and `.well-known/agent.json` as a generic root pointer. The unresolved operational question is whether merchants will need a single manifest serving all agent types, two or three complementary manifests, or a separate manifest for each ecosystem they need to be visible in. Movement at the W3C and GS1 levels suggests some convergence around commerce-specific structured data is underway, but adoption pressure will set the pace as much as formal specification work.

Discovery mechanism. The location at `/.well-known/lar.json` does not, by itself, render the manifest discoverable: agents must know the filename in advance. For the convention to

reach agents without prior framework knowledge, the manifest can be referenced from surfaces agents already scan. Three complementary mechanisms are available: a `LAR:` directive in `robots.txt` (analogous to the existing `Sitemap:` convention), a `<link rel="lar">` in the HTML head of rendered pages (aligned with adjacent proposals such as Vercel's `<script type="text/llms.txt">` and Marro et al.'s `<link rel="agent-permissions">`), and a cross-reference from `/llms.txt` where present. Each mechanism reaches a different class of agent (crawler-class, HTML-rendering, llms.txt-aware); empirical observation will inform which gains traction. Discovery mechanism is open for community refinement.

Layered architecture vocabulary. The terminology of "layered architecture" for agent systems has accumulated several distinct uses. Huang's seven-layer reference architecture (Springer 2025) decomposes the internal agent stack from foundation models to ecosystem deployment.[25] Fleming et al. (2025) propose Agent Communication (L8) and Agent Semantic (L9) layers as protocol extensions above OSI transport.[26] Liang's seven-layer compute model (2025) frames AI computation across hardware-to-application strata.[27] LAR addresses a different scope: publisher-side institutional declarations for agents arriving without prior context, organised by workflow layer rather than by infrastructure stratum or protocol hierarchy. The shared vocabulary reflects a broader recognition that agentic systems require multiple coordinated layers; the layers in question differ in nature.

Trust questions.

Liability and provenance. A versioned, signed, timestamped LAR surface is a legal artifact in a way that scraped HTML is not. The EU AI Act's Article 50 imposes transparency obligations on providers and deployers of certain AI systems — including any agent-style system that interacts directly with users or generates synthetic content — and becomes applicable on 2 August 2026. Italy's Legge 34/2026 (Articles 18–23, in force 7 April 2026) establishes lawfulness criteria, traceability requirements, and a removal regime for online reviews of restaurants, hotels, and tourist attractions, enforced by AGCM and AGCOM; while its scope is restaurant- and tourism-specific, it is an early instance of statutory obligations attaching to the publishing layer of consumer-facing platforms. The U.S. FTC's Operation AI Comply, launched 25 September 2024, has applied Section 5 authority to AI-mediated commercial claims. None of these regimes mandates LAR specifically; what they raise is the question of what evidence a merchant can produce of what they actually authorised an agent to act upon. The legal distinction between *declaring* a surface and *contractually attesting to* a transaction at an endpoint is not yet tested in case law. The technical infrastructure for signing and versioning LAR surfaces exists (Web Bot Auth, HTTP Message Signatures);[28] the legal frameworks operating on top are early.

Revocation under loss of control. A merchant who signs a surface and later loses control of the domain, signing key, or publishing infrastructure cannot revoke the surface by publishing a new one — the channel for revocation has been compromised along with the channel that needed revoking. The established remedies (CRLs, OCSP, key revocation registries, Certificate Transparency append-only logs) all rely on a channel separate from the one whose validity is being revoked. Where the out-of-band channel for LAR lives, who maintains it, and how an agent verifying a surface knows to consult it are questions the architecture does not currently answer.

Verification at the leaf. A manifest can be deterministic without the content at its leaves being trustworthy. An agent that walks a LAR surface to a leaf containing fabricated,

outdated, or unverified claims still ends with a bad decision. Anthropic's Project Deal (April 2026), which observed Claude agents negotiating on behalf of human principals, documented agents confabulating personal details about their principals during extended exchanges; the authors note explicitly that such confabulations "illustrate the potential risks of implementing a system like this in a non-experimental setting without additional safeguards."[29] The architectural question of how to sign and version what a LAR surface publishes is therefore inseparable from the editorial question of what the agent reading it is permitted to add, omit, or invent on its principal's behalf.

Runtime questions.

State management at the endpoint. LAR is a publishing pattern; it does not specify the runtime semantics of the endpoints it declares. High-velocity commerce introduces concerns the publishing layer does not address — quote generation with finite validity, inventory locks during agent deliberation, idempotency tokens for retried transactions, session continuity across multi-step workflows. An agent that reads `pricing.json` at one moment and submits a transaction fifteen minutes later may face flash-sale expiry, stock depletion, or policy revision in the interval. These are protocol-level concerns (UCP, ACP, MCP, or proprietary commerce protocols handle quote and lock semantics) but their integration with LAR-declared endpoints — how a surface declares which endpoints provide quote semantics, what freshness windows agents can rely on, when re-reading the surface is required — is not formalised. This is among the most consequential unresolved areas for production e-commerce adoption. Agent-side behavioural questions follow from this: how an agent detects a stale or incomplete surface, when it falls back to direct endpoint queries or to alternative sources, and how it signals reduced confidence to its principal — are equally unresolved. The reference examples adopt a working `as_of` + `freshness_window_minutes` convention on every volatile file; whether this becomes the canonical declaration mechanism remains open for community refinement.

9. Conclusion

The web is entering a transitional phase in which human-facing interfaces increasingly coexist with parallel machine-legible surfaces designed for autonomous systems. The forms of these surfaces remain unsettled, fragmented, and vendor-specific. Yet across heterogeneous ecosystems, similar structural responses are emerging: manifest-driven discovery, progressive disclosure, workflow-oriented organization, and deterministic machine-readable retrieval.

This paper proposes Layered Agentic Retrieval (LAR) as a conceptual vocabulary for describing that transition. The pattern's architectural commitments — plain text, progressive disclosure, workflow-organised hierarchy — distinguish it from both retrofit and from probabilistic retrieval; its lineage in RESTful hypermedia and the Semantic Web situates it within decades of prior work on agent-readable resources.

Whether it converges on a single canonical shape, several adjacent shapes, or fragments further is an empirical question whose answer will be set by adoption and by the empirical research the open questions above invite. The terminology proposed here is offered to operators who need shared vocabulary now, regardless of whether formal specification follows.

The convention proposed in this paper is intentionally minimal in its v1 articulation. The canonical filename (`/.well-known/lar.json`), the JSON Schema for the manifest, the discovery mechanism, and several other operational details are deliberately left open for community input. The project repository at github.com/lar-spec/lar hosts ongoing discussion; refinement consensus reached during the open review period will be integrated into subsequent versions of this paper. Empirical validation of these commitments is planned in partnership with Italian mid-market merchants (Marinoni Moretto & Antichi, forthcoming 2026).

Author's Note & Disclosures

This paper develops material that appears, in different form, in an appendix to the forthcoming book *Selling to Agents: The Merchant Playbook for Agentic Commerce* by the same author. The standalone version published here is released under CC BY-SA 4.0; the book version is under standard commercial copyright. The two are not identical: this paper is structured for academic and practitioner discussion, with adapted introduction, abstract, and citation form.

Notes

1. OpenAI Platform Documentation, /codex/llms-full.txt, accessed April 2026.
2. Google Agent Development Kit Documentation, llms-full export, accessed April 2026.
3. Apideck Developer Documentation, llms.txt + llms-full.txt + md/index.json structure, accessed April 2026.
4. Howard, J. (2024) "The /llms.txt file" specification proposal. Production implementations surveyed include Meteor, FastMCP, assistant-ui, Chkk, Mintlify.
5. Anthropic Claude Skills documentation, accessed April 2026. The "progressive knowledge disclosure" framing is documented in Claude Skill Design materials and the broader MCP ecosystem.
6. Throughout this paper, surface refers to the publication artefact an institution exposes for agent consumption — the structured, machine-readable representation of what is offered, under which terms, with which capabilities for interaction, rendered machine-legible by workflow-organised content that an agent can act upon. The concept is independent of the canonical filename: the architecture of the surface (root manifest with linked deeper material, format-follows-purpose, workflow-organised hierarchy) is invariant whether the canonical file is named `lar.json`, `agent-surface.json`, or another community-determined alternative within the well-known namespace. This usage of "surface" is also distinct from adjacent senses of the term established in agent-payment protocols, where it denotes a UI role (e.g., AP2's "Trusted Surface" as the interface trusted to obtain user consent for transaction mandates, or "agent surface" as in OpenAI/Google parlance for consumer-facing AI assistants). The two usages are complementary rather than competing: a publication surface (LAR) is what an institution exposes; a transaction surface (AP2/UCP/ACP terminology) is where an agent acting on a principal's behalf interacts with the user. Both senses share the underlying intuition of "surface" as a thin layer of declared interaction, but at different levels of the agent-mediated commerce stack. A separate practitioner project at agentsurface.dev (Howells, 2026) uses "Agent Surface" as a scoring framework for codebase agent-readability; its scope (audit of existing codebases) is distinct from the publisher-side institutional declaration scope of LAR.
7. Microsoft NLWeb initiative, accessed April 2026.
8. Universal Commerce Protocol (UCP), Google + retail coalition, announced 11 January 2026 at NRF. Co-developed with Shopify, Etsy, Wayfair, Target, Walmart; endorsed by 20+ additional partners. Specification at ucp.dev. Compatible with Agent Payments Protocol (AP2), Agent2Agent (A2A), and Model Context Protocol (MCP).
9. Agentic Commerce Protocol (ACP), OpenAI + Stripe, announced 29 September 2025. Apache 2.0 license; specification at agenticcommerce.dev. Uses Shared Payment Tokens for credential-free agent-mediated

transactions. Acronym disambiguation: this paper uses "ACP" exclusively for the OpenAI/Stripe commerce protocol; an unrelated protocol with the same acronym (IBM BeeAI's Agent Communication Protocol) has been formally merged into Google's Agent2Agent (A2A) under the Linux Foundation umbrella.

10. WebMCP, W3C Web ML Community Group Draft Community Group Report (latest draft 10 February 2026). Early 2026 Chrome 146 preview builds exposed WebMCP behind the "Experimental Web Platform Features" flag, with subsequent Edge experimentation reported in ecosystem documentation. Defines `navigator.modelContext` as a browser-native API where the page itself acts as a tool surface for in-browser agents — distinct from Anthropic's server-side MCP. As a Community Group Report, WebMCP is not on the W3C Recommendation track.
11. Google, "Optimizing your website for generative AI features on Google Search." Search Central documentation, last updated 15 May 2026. Available at: https://developers.google.com/search/docs/fundamentals/ai-optimization-guide. Quoted passages from "Mythbusting" section (subsection "LLMS.txt files and other 'special' markup") and "Explore agentic experiences" section.
12. Jesus, A. & Morrison, V. (2026). "Measure your website's agent readiness, score from 0 to 100, and improve it." Cloudflare Blog, 17 April 2026. Available at: https://blog.cloudflare.com/agent-readiness/. The Agent Readiness scoring framework and its four-dimension methodology are documented in the blog post; the live measurement dashboard at https://radar.cloudflare.com/ai-insights publishes weekly-updated adoption data; underlying definitions and methodology are in the Cloudflare Radar Glossary, https://developers.cloudflare.com/radar/glossary/. The 108,595 domains figure cited in the body text is a dashboard snapshot reading from May 2026 and is time-sensitive; subsequent values are accessible at the live dashboard.
13. MajesticLabs, hierarchical-agents skill, 2026 (LobeHub Skills Marketplace). Generates root and sub-folder AGENTS.md files with nearest-wins resolution and just-in-time indexing.
14. Empirical validation of the context-window economics motivating workflow-organised structure has accumulated since this paper was drafted. Pinecone reports in its KRAFTBench/Nexus launch materials (Zhu & Ragavan, "Better Models Won't Save Your Agent," Pinecone Blog, 4 May 2026, https://www.pinecone.io/blog/introducing-nexus-knowledge-engine/), comparing three retrieval strategies on 150 questions over 493 S&P 500 10-K filings, that compiled-knowledge artifacts reduced average token use from 49,103 (agentic RAG) and 528,301 (coding-agent baseline) to 6,733 — a 7× and 80× multiplier respectively — alongside latency and task-completion gains. These are vendor benchmark results, not independent replication. The benchmark targets consumer-side enterprise compilation (Pinecone Nexus knowledge engine), not publisher-side declaration, but the underlying economic argument — that pre-compiled, workflow-shaped knowledge structures collapse token cost by orders of magnitude relative to retrieval-time assembly — applies symmetrically across both sides of the agent's information boundary. Microsoft (Fabric IQ, FabCon March 2026), Google (Knowledge Catalog, April 2026), and Pinecone (Nexus, May 2026) shipped enterprise consumer-side knowledge compilation infrastructure during this paper's final preparation, confirming the broader category while operating on a different architectural layer than LAR's publisher-side scope.
15. The distinction: RAG performs probabilistic retrieval over an unstructured or weakly-structured corpus to provide context for generation. LAR is the publishing pattern that creates the conditions for deterministic retrieval by structuring the source. The two are complementary at the system level — an agent might use RAG over a weakly-structured corpus and direct LAR retrieval over a well-structured one in the same workflow — but they are categorically distinct as architectural commitments of the publisher.
16. Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine. Available at: https://ics.uci.edu/~fielding/pubs/dissertation/top.htm.
17. Berners-Lee, T., Hendler, J., & Lassila, O. (2001). "The Semantic Web." Scientific American, 284(5), 34–43. DOI: 10.1038/scientificamerican0501-34. Available at: https://www.scientificamerican.com/article/the-semantic-web/.
18. Karpathy, A. (2026) "LLM Knowledge Bases," tweet of 3 April 2026, followed by GitHub Gist llm-wiki.md (archived at https://web.archive.org/web/2026*/https://gist.github.com/karpathy/442a6bf555914893e9891c11519de94f). Three-layer architecture: raw/ (immutable user sources) + wiki/ (LLM-maintained markdown pages) +

CLAUDE.md or AGENTS.md (schema). The pattern lineage cited by Karpathy is Vannevar Bush's Memex (1945).

19. Volpini, A., Raad, E., Gamba, B., & Riccitelli, D. (2026) "Structured Linked Data as a Memory Layer for Agent-Orchestrated Retrieval." arXiv preprint 2603.10700v1, March 2026. Available at: https://arxiv.org/abs/2603.10700. The +29.6% retrieval accuracy improvement is measured across editorial, legal, travel, and e-commerce domains using Vertex AI Vector Search 2.0 for retrieval and the Google Agent Development Kit (ADK) for agentic reasoning. Ground-truth answers are derived from the same knowledge graph used for enhancement, a methodological caveat the authors disclose.
20. Kim, E. Y. & Huang, J. (2026). "FinRetrieval: A Benchmark for Financial Data Retrieval by AI Agents." arXiv preprint 2603.04403, January 2026. Available at: https://arxiv.org/abs/2603.04403. Reports a 71 percentage point gap (90.8% versus 19.8%) for Claude Opus 4.5 accessing structured financial data via Model Context Protocol versus web search alone, across 500 financial retrieval questions and 14 agent configurations spanning Anthropic, OpenAI, and Google frontier models. Authors are employed by Daloopa (disclosed in Section 7.3 of the cited paper). The paper decomposes the headline gap into approximately equal behavioural and infrastructure components (Section 5.3, Finding A). The scope differs from Volpini et al. (note 19) — measuring tool-mediated structured access rather than publishing format — but the directional finding (structured retrieval substantially outperforming unstructured retrieval for agents) is supported across both layers.
21. Interpretive framing from pre-publication review: Andrea Volpini (WordLift) articulated the relationship between enhanced entity pages and LAR as peer projections of a single canonical knowledge graph — by entity and by workflow respectively. This is interpretive input from a technical reviewer with adjacent published work, not external published evidence. Disclosure: Volpini and his team at WordLift authored the controlled experiment cited at note 19; no commercial relationship between this author and WordLift; integration of Volpini's framing into this work was at the author's editorial discretion.
22. Pelletteria Vanelli, the composite Florentine leather goods merchant referenced throughout §6, is constructed for illustrative purposes, drawing on observable patterns across artisanal Italian e-commerce (third-generation family operations, sub-100 SKU catalogs, premium artisanal positioning, in-house digital teams) without representing any single existing business. Composition rather than direct case study allows architectural illustration without exposing a specific merchant's operational data or commercial relationships.
23. Zhou, S. et al. (2023). "WebArena: A Realistic Web Environment for Building Autonomous Agents." arXiv:2307.13854. Available at: https://arxiv.org/abs/2307.13854. Reports a 14.41% end-to-end task success rate for the best GPT-4-based agent against 78.24% human performance on 812 long-horizon tasks. Deng, X. et al. (2023). "Mind2Web: Towards a Generalist Agent for the Web." NeurIPS 2023 Datasets and Benchmarks Track. Available at: https://arxiv.org/abs/2306.06070. Comprises 2,000+ open-ended tasks across 137 websites in 31 domains. Both benchmarks evaluate agent performance over agent-naïve human-facing surfaces; neither evaluates against agent-native publishing alternatives.
24. Marinoni Moretto, F., & Antichi, F. (Forthcoming 2026). *Empirical Benchmarking of Layered Agentic Retrieval (LAR) Surfaces in Mid-Market E-commerce Environments*.
25. Huang, K. & Huang, J. (2025). "AI Agent Tools and Frameworks." In Huang, K. (Ed.), *Agentic AI*. Springer (Progress in IS series), pp. 23–50. DOI: 10.1007/978-3-031-90026-6_2. Available at: https://link.springer.com/chapter/10.1007/978-3-031-90026-6_2. The seven-layer reference architecture decomposes the agent stack from Foundation Models (Layer 1) through Data Operations, Agent Frameworks, Deployment Infrastructure, Evaluation, Security, to Agent Ecosystem (Layer 7). Adopted by the Cloud Security Alliance as the basis of the MAESTRO threat-modeling framework.
26. Fleming, C., Pandey, V., Kompella, R., & Muscariello, L. (2025). "A Layered Protocol Architecture for the Internet of Agents." arXiv preprint 2511.19699. Available at: https://arxiv.org/abs/2511.19699. Proposes Agent Communication Layer (L8) and Agent Semantic Layer (L9) as additional layers above the OSI/TCP-IP application transport. L8 standardises message envelopes, speech-act performatives, and interaction patterns; L9 establishes shared semantic context discovery, negotiation, and grounding before task execution.
27. Liang, B.-S. (2025). "AI Compute Architecture and Evolution Trends." arXiv preprint 2508.21394. Available at: https://arxiv.org/abs/2508.21394. Proposes a seven-layer model spanning Physical Layer, Link Layer, Neural Network Layer, Context Layer, Agent Layer, Orchestrator Layer, and Application Layer for AI compute infrastructure.

28. Web Bot Auth (IETF Internet-Draft, available at: https://datatracker.ietf.org/doc/draft-meunier-web-bot-auth-architecture/) and HTTP Message Signatures (RFC 9421, available at: https://www.rfc-editor.org/rfc/rfc9421) provide the cryptographic primitives. Their integration with publisher liability frameworks is an open research question across the broader literature on AI commerce regulation.
29. Troy, K., Shields, D., Bradwell, K., & McCrory, P. (2026). "Project Deal: Our Claude-run Marketplace Experiment." Anthropic, posted April 24, 2026. Available at: https://www.anthropic.com/features/project-deal. The quoted sentence is drawn from footnote 14 of the published study. As a publishing artifact, the page is also illustrative of the retrieval and verification friction this paper addresses: citation metadata and links to underlying materials are nested within a long-form human-readable page, while the linked PDFs are served from a CDN whose robots.txt disallows PDF crawling (Disallow: /*.pdf; Disallow: /*.PDF). For compliant agents, this increases the traversal cost of primary-source verification and citation relative to a dedicated machine-legible publication surface.

Appendix A: Initial Schema Proposal (v0.1 Draft)

A.1 JSON Schema for Canonical Manifest

The following schema defines the structural requirements for the `/.well-known/lar.json` discovery document.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Layered Agentic Retrieval (LAR) Manifest",
  "description": "Root discovery document declaring agent-navigable surfaces, capabilities, and
institutional attestations.",
  "type": "object",
  "required": ["lar_version", "publisher", "identity", "current_state", "operations"],
  "properties": {
    "lar_version": {
      "type": "string",
      "description": "Version of the LAR specification (e.g., '0.1')."
    },
    "publisher": {
      "type": "object",
      "required": ["name", "domain"],
      "properties": {
        "name": { "type": "string" },
        "domain": { "type": "string", "format": "hostname" },
        "legal_entity_id": { "type": "string", "description": "Optional LEI or local tax
identifier for compliance." }
      }
    },
    "identity": {
      "type": "string",
      "format": "uri-reference",
      "description": "URI to the root Markdown document containing institutional identity,
brand context, and narrative."
    },
    "catalog": {
      "type": "string",
      "format": "uri-reference",
      "description": "URI to the static product/service catalog tree."
    },
    "current_state": {
      "type": "object",
      "description": "Pointers to volatile state endpoints or files.",
      "properties": {
        "availability": { "type": "string", "format": "uri-reference" },
        "pricing": { "type": "string", "format": "uri-reference" }
```

```
      }
    },
    "operations": {
      "type": "object",
      "description": "Declared transactional endpoints authorized by the publisher.",
      "patternProperties": {
        "^[a-zA-Z0-9_-]+$": {
          "type": "object",
          "required": ["endpoint", "protocol"],
          "properties": {
            "endpoint": { "type": "string", "format": "uri-reference" },
            "protocol": {
              "type": "string",
              "enum": ["ACP", "UCP", "MCP", "WebMCP", "A2A", "OpenAPI", "Proprietary",
"Other"],
              "description": "The execution protocol the endpoint expects."
            },
            "spec": {
              "type": "string",
              "format": "uri-reference",
              "description": "Optional URI to OpenAPI/Swagger spec detailing the endpoint
payload."
            }
          }
        }
      }
    },
    "policies": {
      "type": "object",
      "description": "URIs to governing policies under which operations are authorized.",
      "properties": {
        "terms": { "type": "string", "format": "uri-reference" },
        "returns": { "type": "string", "format": "uri-reference" },
        "privacy": { "type": "string", "format": "uri-reference" }
      }
    },
    "attestation": {
      "type": ["object", "null"],
      "description": "Cryptographic signature verifying the integrity and authorization of the
manifest and its linked tree.",
      "properties": {
        "method": { "type": "string", "enum": ["WebBotAuth", "HTTPMessageSignature"] },
        "timestamp": { "type": "string", "format": "date-time" },
        "signature": { "type": "string" },
        "public_key_uri": { "type": "string", "format": "uri" }
      }
    }
  }
}
```

A.2 Identity Template (about.md)

The structural template ensuring token-efficient institutional context separation.

```
# [Institution Name] - Institutional Identity & Context
**LAR Surface Version:** [Version]
**Last Updated:** [ISO-8601 Date]
**Canonical Manifest:** [URL to lar.json]

## 1. Core Identity
*   **Legal Name:** [Full Legal Name]
*   **Operating Name:** [DBA / Brand Name]
*   **Primary Jurisdiction:** [Country/State]
*   **Registration/Tax ID:** [ID Number]
*   **Primary Domain:** [Domain]
*   **Contact for Agent Operators:** [Dedicated Email/Endpoint]

## 2. Institutional Context (Narrative)
*Note for Agent: This section provides semantic context for brand positioning, target audience,
and quality standards. Use this to calibrate conversational tone or product recommendations.*

[2-3 paragraphs of dense, high-signal prose explaining the institution's history, value
proposition, and market positioning. NO marketing filler.]

## 3. Operational Guarantees & Certifications
*   **Certifications:** [e.g., ISO 9001, B-Corp, Fair Trade]
*   **Manufacturing/Sourcing Origin:** [e.g., 100% made in Italy]
*   **Quality Standard:** [Brief description of warranty or quality floor]

## 4. Policy Pointers
*Note for Agent: Do not hallucinate policies. Always resolve the following endpoints for
deterministic policy constraints prior to generating a transaction quote.*
*   **Terms of Service:** [Relative link to terms.md]
*   **Shipping & Fulfillment:** [Relative link to shipping.md]
*   **Returns & Refunds:** [Relative link to returns.md]
```