Individual Submission
Internet-Draft
Intended status: Informational
Expires: 25 October 2026

K. Tsoi
Aevum Network

23 April 2026

AgentCard: A Framework-Neutral Identity and Capability Declaration Format for Agent-to-Agent Communication

draft-aevum-agentcard-00

Abstract

This document defines AgentCard, a lightweight JSON data format that enables autonomous software agents to declare their identity, capabilities, communication endpoints, and resource pricing in a framework-neutral manner. AgentCard is designed for agent-to-agent (A2A) communication scenarios where agents built with different frameworks — such as LangChain, CrewAI, AutoGen, or custom implementations — must interoperate without prior coordination.

An AgentCard is a pure data schema: it carries no execution logic and imposes no transport requirements. It is analogous to an HTTP response header set — a machine-parseable self-description that any compliant reader can interpret.

Key properties of AgentCard include a globally unique ULID-based agent identity, dot-namespaced capability identifiers compatible with OpenAI function-calling and Model Context Protocol (MCP) tool schemas, a physics-grounded energy pricing floor derived from Landauer’s principle, and an extensible metadata namespace for framework-specific annotations.

Discussion Venues

Discussion of this document should take place on the GitHub repository at https://github.com/kwailapt/AgentCard/issues. The JSON Schema for AgentCard v1.0 is maintained at https://github.com/kwailapt/AgentCard/blob/main/schema.json.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The proliferation of autonomous AI agent frameworks has created a fragmentation problem: an agent built with LangChain cannot natively describe its capabilities to an AutoGen orchestrator; a CrewAI researcher agent cannot publish its endpoint in a form that a custom MCP server can consume without bespoke integration code. Each framework has invented its own internal identity representation, none of which is interoperable by default.

This situation is analogous to the pre-HTTP web, where each networked application defined its own message envelope format. The solution was not to mandate a single application protocol but to define a universal, lightweight header format (HTTP headers) that any application could attach to its messages.

AgentCard is the equivalent of HTTP headers for the agent web. It defines a minimal JSON document — the "card" — that any agent can produce and any peer can consume, independent of the agent's internal implementation, programming language, or hosting environment.

1.1. Design Goals

1. *Framework neutrality.* An AgentCard produced by a LangChain tool MUST be interpretable by an AutoGen registry and vice versa. The format imposes no dependency on any specific agent framework.

2. *Zero execution logic.* AgentCard is a pure data schema. It describes what an agent can do and how to reach it; it does not specify how the agent processes requests. Execution semantics are left entirely to the agent and its transport layer.

3. *Globally unique, coordination-free identity.* Agent identities MUST be globally unique without requiring a central registration authority. This document specifies the use of Universally Unique Lexicographically Sortable Identifiers (ULIDs) [ULID-SPEC] for this purpose.

4. *Physically grounded resource accounting.* AI computation has non-zero energy cost. AgentCard includes an optional pricing field whose minimum value is constrained by Landauer's principle [LANDAUER], preventing fraudulent "zero-cost" capability claims in multi-agent economic systems.

5. *Extensibility without breakage.* New fields MAY be added to future versions of AgentCard. Readers MUST ignore unknown fields. The metadata object provides a namespaced extension point for framework- and application- specific annotations.

1.2. Non-Goals

AgentCard does not specify:

* How agents authenticate to each other at the transport layer.

* The wire format for agent-to-agent message exchange.

* How capability invocations are dispatched or scheduled.

* Agent lifecycle management (creation, termination, migration).

* Trust establishment or public-key infrastructure for agents.

These concerns are addressed by complementary protocols (e.g., OAuth 2.0 [RFC6749], WIMSE [I-D.ietf-wimse-arch], and MCP [MCP-SPEC]).

2. The AgentCard Data Model

An AgentCard is a JSON object [RFC8259] containing the fields defined in this section. The fields agent_id, name, version, capabilities, and endpoint are REQUIRED. All other fields are OPTIONAL.

The normative JSON Schema for AgentCard v1.0, expressed in JSON Schema draft 2020-12 [JSON-SCHEMA], is maintained at the reference implementation repository [AGENTCARD-SPEC].

2.1. agent_id

Type: string

Required: yes

Constraints: Exactly 26 characters. Each character MUST be a member of the Crockford Base32 alphabet: digits 09 and uppercase letters AZ excluding I, L, O, and U.

The agent_id field carries the agent’s globally unique identity as a ULID [ULID-SPEC]. A ULID encodes a 48-bit millisecond-precision timestamp and 80 bits of cryptographically random data into 128 bits total, then serialises the result as a 26-character Crockford Base32 string.

ULIDs provide the following properties relevant to agent identity:

* *Global uniqueness without coordination.* The 80-bit random component gives a collision probability of approximately 2^{-80} per pair of independently generated identities.

* *Lexicographic time-ordering.* Agents created later sort after agents created earlier under standard string comparison, which simplifies registry indexing.

* *URL-safety.* The Crockford Base32 alphabet contains no characters that require percent-encoding in URIs.

An agent MUST generate its agent_id using a cryptographically secure random number generator for the random component of the ULID. Predictable or sequential random components are PROHIBITED.

Example:

```
"agent_id": "01HZQK3P8EMXR9V7T5N2W4J6C0"
```

2.2. name

Type: string

Required: yes

Constraints: 1 to 128 Unicode code points. MUST NOT be empty.

A human-readable display name for the agent. The name field is intended for presentation purposes and SHOULD reflect the agent's primary function or role. It is NOT required to be globally unique; uniqueness is provided by agent_id.

```
"name": "WebSearchAgent"
```

2.3. version

Type: string

Required: yes

Constraints: MUST conform to Semantic Versioning 2.0.0 [SEMVER].

The version of this AgentCard document. Implementations SHOULD increment the PATCH component when reissuing an AgentCard with corrected metadata. They SHOULD increment the MINOR component when new capabilities are added in a backward-compatible manner. They SHOULD increment the MAJOR component when capabilities are removed or endpoint URLs change incompatibly.

```
"version": "1.0.0"
```

2.4. capabilities

Type: array of objects

Required: yes

Constraints: MUST contain at least one entry.

An ordered array of capability declarations. Each entry describes one thing the agent can do. Entries SHOULD be ordered from most prominent to least prominent capability.

2.4.1. capabilities[].id

Type: string

Required: yes

Pattern: ^[a-z0-9][a-z0-9._-]*$

A dot-namespaced capability identifier. The identifier MUST begin with a lowercase letter or digit. Subsequent characters MAY be lowercase letters, digits, dots, underscores, or hyphens.

Convention for the dot-namespace:

* text.* — natural language processing capabilities (e.g., text.generate, text.summarise)
* tool.* — executable tool wrappers (e.g., tool.web_search, tool.python_repl)
* data.* — data retrieval and transformation (e.g., data.query_sql, data.fetch_csv)
* fn.* — named function exposure compatible with OpenAI function-calling schema (e.g., fn.get_weather)
* a2a.* — agent coordination capabilities (e.g., a2a.delegate, a2a.negotiate)

Implementations SHOULD use the conventional namespaces above. Private or experimental namespaces MAY use a reverse-domain prefix (e.g., com.example.custom_capability).

The capabilities[].id is designed to be compatible with the name field of an OpenAI-style function schema and the name field of an MCP tool definition, enabling zero-conversion interoperability.

2.4.2. capabilities[].description

Type: string

Required: no (RECOMMENDED)

A human-readable description of the capability for use in LLM prompts, API documentation, and agent discovery UIs. Implementations SHOULD provide this field. The description SHOULD be one to three sentences that describe what the capability does, what inputs it accepts, and what it returns.

2.4.3. capabilities[].input_schema and capabilities[].output_schema

Type: object (JSON Schema)

Required: no

Optional JSON Schema objects describing the expected input parameters and output structure of the capability. When provided, these schemas MUST conform to JSON Schema [JSON-SCHEMA] and SHOULD be compatible with the parameters object in OpenAI function-calling format.

2.4.4. capabilities[].tags

Type: array of strings

Required: no

Optional list of free-form tags for discovery and filtering.

2.4.5. Example

Example capabilities array:

```
"capabilities": [
  {
    "id": "text.summarise",
    "description": "Summarise a document to a target length.",
    "input_schema": {
      "type": "object",
      "properties": {
        "text":       { "type": "string" },
        "max_words":  { "type": "integer", "default": 200 }
      },
      "required": ["text"]
    }
  },
  {
    "id": "tool.web_search",
    "description": "Search the public web and return top results.",
    "tags": ["search", "retrieval"]
  }
]
```

2.5. endpoint

Type: object

Required: yes

Describes how to reach the agent. Contains two REQUIRED sub-fields and one OPTIONAL sub-field:

2.5.1. endpoint.protocol

REQUIRED. MUST be one of: http, https, grpc, stdio, or mcp.

The value mcp indicates that the agent is accessible via the Model Context Protocol [MCP-SPEC]. The value stdio indicates an agent that communicates over standard input/output streams (applicable to local subprocess agents).

2.5.2. endpoint.url

REQUIRED. A URI [RFC3986] identifying the agent's communication endpoint. The URI MUST be consistent with the declared protocol (e.g., an https endpoint MUST begin with the scheme https://).

2.5.3. endpoint.auth

OPTIONAL. An object describing the authentication scheme required to contact the endpoint. The auth.scheme sub-field MUST be one of: none, bearer, api_key, oauth2, or mtls. The default value is none.

Presence of an auth object does not constitute credential exchange; it is a declaration of the authentication mechanism required. Actual credential negotiation is handled by the transport layer.

2.5.4. Example

```
"endpoint": {
  "protocol": "https",
  "url": "https://agents.example.com/api/summariser",
  "auth": { "scheme": "bearer" }
}
```

2.6. pricing

Type: object

Required: no

Describes the resource cost of invoking the agent. All cost values are expressed in SI units: joules for energy.

2.6.1. pricing.base_cost_joules

The minimum energy cost per invocation in joules. If present, this value MUST be either exactly zero (the agent imposes no energy charge) or greater than or equal to the Landauer limit at 300 K:

L = k_B * T * ln(2) = 1.381e-23 J/K * 300 K * 0.693 = 2.854e-21 J

where k_B is the Boltzmann constant (1.380649 × 10^{-23} J/K, SI 2019 definition) and T is the ambient temperature in kelvin.

Landauer's principle [LANDAUER] establishes that erasing one bit of information in a system at temperature T dissipates at least k_B * T * ln(2) joules as heat. Any real computation that modifies memory must erase at least one bit; therefore no physical computation can have a base energy cost strictly between zero and the Landauer limit. A base_cost_joules value in the open interval (0, 2.854e-21) is physically impossible and MUST be rejected by conformant validators.

This constraint is not a billing requirement: it is a physical plausibility check. Agents SHOULD set base_cost_joules to a realistic estimate of their per-invocation energy consumption. This enables multi-agent economic systems to perform energy-accounting without fraudulent zero-cost claims.

2.6.2. pricing.per_token_joules

The additional energy cost per output token, in joules. If present, MUST be non-negative.

2.6.3. Example

```
"pricing": {
  "base_cost_joules": 2.854e-21,
  "per_token_joules": 1.4e-24
}
```

2.7. metadata

Type: object

Required: no

An open-ended object for framework- and application-specific annotations. Implementations MUST NOT reject an AgentCard because it contains unrecognised metadata keys.

Keys beginning with the prefix pacr: are reserved for annotations derived from Physically Annotated Causal Records (PACR) as defined in the Aevum Network specification. Keys beginning with the prefix mcp: are reserved for Model Context Protocol annotations. All other prefixes are unregistered and available for private use.

The following pacr:-prefixed keys are defined in this document:

pacr:trust_tier A trust classification derived from the agent's causal return rate. MUST be one of: untrusted, basic, established, verified, or banned. Absence of this field implies untrusted.

pacr:substrate_scope A free-form string identifying the computational substrate (hardware or data population) on which the agent was trained or calibrated (e.g., "M1_Ultra", "UKBiobank"). This field is used to prevent data-bias ossification in multi-agent systems that aggregate statistics across heterogeneous agents.

```
"metadata": {
  "pacr:trust_tier": "established",
  "pacr:substrate_scope": "AWS_Graviton_c7g",
  "framework": "crewai",
  "created_at": "2026-04-23T00:00:00Z"
}
```

2.8. goal_subscriptions

Type: array of objects

Required: no

An optional list of shared goals that this agent has subscribed to participate in. This field supports multi-agent coalition formation: an orchestrator can discover all agents that have subscribed to a given goal identifier and assemble a coalition without prior explicit configuration.

Each entry contains a REQUIRED goal_id string, an OPTIONAL human-readable description, and an OPTIONAL priority value in the range [0, 1] indicating the agent's relative commitment to this goal.

```
"goal_subscriptions": [
  {
    "goal_id": "01HZQK3P8EMXR9V7T5N2W4J6C1",
    "description": "Competitive analysis for Q3 2026",
    "priority": 0.8
  }
]
```

3. Serialization and Media Type

An AgentCard MUST be serialised as a JSON object [RFC8259]. The canonical serialisation MUST use UTF-8 character encoding [RFC3629].

The RECOMMENDED media type for AgentCard documents is application/agentcard+json (see Section 9.1). Implementations SHOULD also accept application/json for backward compatibility.

When an AgentCard is embedded in another document (e.g., an MCP tool response), the embedding format MAY serialize the AgentCard as a JSON string containing an escaped JSON object. Implementations MUST accept both embedded-string and direct- object forms.

4. Well-Known URI Discovery

An agent that serves its AgentCard at a well-known HTTP(S) endpoint SHOULD publish it at the path /.well-known/agentcard relative to the agent’s base URL (see Section 9.2).

For example, an agent at https://agents.example.com/ SHOULD publish its AgentCard at:

```
GET https://agents.example.com/.well-known/agentcard
Accept: application/agentcard+json

HTTP/1.1 200 OK
Content-Type: application/agentcard+json
Cache-Control: max-age=3600

{
  "agent_id": "01HZQK3P8EMXR9V7T5N2W4J6C0",
  "name": "ExampleAgent",
  ...
}
```

The response SHOULD include a Cache-Control header. AgentCards are relatively stable documents; a max-age of 3600 seconds (one hour) is RECOMMENDED. Validators and registries SHOULD re-fetch at the cache expiry interval.

Agents that use the mcp protocol SHOULD also expose their AgentCard via the AgentCard MCP server tools defined in [AGENTCARD-MCP].

5. Validation Rules

A conformant AgentCard validator MUST enforce the following rules. A document that fails any MUST rule is invalid and MUST be rejected.

1. agent_id MUST be exactly 26 characters drawn from the Crockford Base32 alphabet ([0-9A-HJKMNP-TV-Z]).

2. version MUST conform to Semantic Versioning 2.0.0. The regular expression is: ^(?:0|[1-9]\d*)\.(?:0|[1-9]\d*)\.(?:0|[1-9]\d*)(?:-[0-9A-Za-z\-.]+)?(?:\+[0-9A-Za-z\-.]+)?$

3. capabilities MUST contain at least one entry.

4. Each capabilities[i].id MUST match ^[a-z0-9][a-z0-9._-]*$.

5. endpoint.protocol MUST be one of: http, https, grpc, stdio, mcp.

6. endpoint.url MUST be a valid URI [RFC3986].

7. If pricing.base_cost_joules is present and non-zero, it MUST be greater than or equal to 2.854 × 10^{-21} joules (the Landauer limit at 300 K).

8. If pricing.per_token_joules is present, it MUST be non-negative.

9. metadata.pacr:trust_tier, if present, MUST be one of: untrusted, basic, established, verified, banned.

10. Validators MUST ignore unknown fields at all levels of the AgentCard hierarchy. Unknown fields MUST NOT cause validation failure.

6. Relation to Existing Work

6.1. OAuth 2.0 and WIMSE

OAuth 2.0 [RFC6749] and the emerging WIMSE framework [I-D.ietf-wimse-arch] address credential issuance, delegation, and workload authentication. AgentCard is complementary: it describes what an already- authenticated agent can do and how to reach it. An implementor SHOULD use WIMSE or OAuth for authentication and AgentCard for capability advertisement.

6.2. Model Context Protocol (MCP)

MCP [MCP-SPEC] defines a protocol for communication between language model hosts and tool servers. AgentCard defines a data format for self-description that MCP servers can publish. The two are orthogonal: an MCP server MAY publish an AgentCard; an AgentCard MAY reference an MCP endpoint.

The capabilities[].id dot-namespace convention in AgentCard is intentionally compatible with MCP tool name fields to minimise conversion overhead.

6.3. Decentralised Identifiers (DIDs)

DID Documents [W3C-DID] provide verifiable, self-sovereign identity using public-key cryptography. AgentCard does not incorporate cryptographic binding by design: requiring agents to manage key pairs introduces deployment friction disproportionate to the use case of framework-neutral capability advertisement. Implementors requiring cryptographic agent identity SHOULD embed a DID reference in the AgentCard metadata field and sign the AgentCard using the DID's verification method.

6.4. OpenAPI and AsyncAPI

OpenAPI [OPENAPI] and AsyncAPI describe the full API surface of a service: paths, parameters, response schemas, authentication flows, and server listings. AgentCard is intentionally lighter: it declares what an agent is and what it can do, not the complete HTTP API surface. AgentCard and OpenAPI are complementary; an AgentCard MAY reference a full OpenAPI document via the metadata field.

7. Complete Example

```
{
  "agent_id": "01HZQK3P8EMXR9V7T5N2W4J6C0",
  "name": "ResearchAnalyst",
  "version": "1.2.0",
  "capabilities": [
    {
      "id": "text.summarise",
      "description": "Summarise a document to a given word limit.",
      "input_schema": {
        "type": "object",
        "properties": {
          "text":      { "type": "string" },
          "max_words": { "type": "integer", "default": 200 }
        },
```

```
        "required": ["text"]
      }
    },
    {
      "id": "tool.web_search",
      "description": "Search the public web and return top-k results.",
      "tags": ["search", "retrieval"]
    },
    {
      "id": "data.fetch_csv",
      "description": "Fetch a CSV file from a URL and return parsed rows."
    }
  ],
  "endpoint": {
    "protocol": "https",
    "url": "https://agents.example.com/api/research-analyst",
    "auth": { "scheme": "bearer" }
  },
  "pricing": {
    "base_cost_joules": 2.854e-21,
    "per_token_joules": 1.4e-24
  },
  "metadata": {
    "pacr:trust_tier": "established",
    "pacr:substrate_scope": "AWS_Graviton_c7g",
    "framework": "crewai",
    "created_at": "2026-04-23T00:00:00Z"
  },
  "goal_subscriptions": [
    {
      "goal_id": "01HZQK3P8EMXR9V7T5N2W4J6C1",
      "description": "Competitive analysis for Q3 2026",
      "priority": 0.8
    }
  ]
}
```

8. Security Considerations

8.1. Capability Spoofing

AgentCard does not authenticate the claims made in the capabilities array. A malicious agent could advertise capabilities it does not possess. Consumers MUST NOT grant elevated privileges or trust solely on the basis of declared capabilities. Operational trust SHOULD be established through observed behaviour over time (e.g., via a causal return rate as described in Section 2.6).

8.2. ULID Collision and Prediction

ULID collision probability is approximately 2^{-80} per pair for randomly generated identifiers. Implementations MUST use a cryptographically secure random number generator for the 80-bit random component. Sequential or low-entropy random components permit identity prediction and MUST NOT be used in adversarial environments.

8.3. Pricing Field Manipulation

The Landauer floor check prevents trivially implausible energy claims but does not prevent an agent from claiming an energy cost far below its true cost. Multi-agent systems that use pricing.base_cost_joules for economic settlement SHOULD independently verify claimed costs against observed latency and throughput.

8.4. Trust Tier Self-Declaration

The metadata.pacr:trust_tier field is self-declared. A malicious agent could claim a high trust tier it has not earned. Consumers SHOULD treat self-declared trust tiers as advisory hints and MUST validate trust tier claims against an independent record of the agent’s interaction history before acting on them.

8.5. Endpoint Redirection

If an agent’s AgentCard is served over HTTP (not HTTPS), an on-path attacker can substitute a malicious endpoint URL. Agents SHOULD serve AgentCards exclusively over TLS (https://). Consumers SHOULD reject AgentCards whose endpoint URL scheme differs from the scheme used to fetch the AgentCard.

8.6. Personally Identifiable Information

AgentCard does not define any fields for personally identifiable information (PII). Implementors MUST NOT embed user data, operator names, or other PII in the name, capabilities[].description, or metadata fields unless the AgentCard is served over a channel with appropriate access controls.

9. IANA Considerations

9.1. Media Type Registration

This document requests that IANA register the following media type in the "Media Types" registry [RFC6838]:

Type name: application

Subtype name: agentcard+json

Required parameters: none

Optional parameters: none

Encoding considerations: binary (UTF-8 JSON object)

Security considerations: See Section 8.

Published specification: This document.

Applications that use this media type: AI agent frameworks, multi-agent orchestrators, MCP servers, and agent registries that implement the AgentCard specification.

Additional information: Magic number(s): none

File extension(s): .agentcard.json

Macintosh file type code(s): none

Person and email address to contact for further information: Kwai Lap Tsoi <kwailapt@gmail.com>

Intended usage: COMMON

Restrictions on usage: none

Author: Kwai Lap Tsoi <kwailapt@gmail.com>

Change controller: IETF

9.2. Well-Known URI Registration

This document requests that IANA register the following well-known URI in the "Well-Known URIs" registry [RFC8615]:

URI suffix: agentcard

Change controller: IETF

Specification document(s): This document.

Related information: The resource at /.well-known/agentcard is an application/agentcard+json document describing the agent accessible at the host.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2018, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/rfc/rfc3629>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/rfc/rfc6838>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/rfc/rfc8615>.

[ULID-SPEC] Feerasta, A., "ULID Specification", 2019, <https://github.com/ulid/spec>.

[SEMVER] Preston-Werner, T., "Semantic Versioning 2.0.0", 2013, <https://semver.org/spec/v2.0.0.html>.

[JSON-SCHEMA] Wright, A., Andrews, H., and B. Hutton, "JSON Schema: A Media Type for Describing JSON Documents (draft 2020-12)", 2022, <https://json-schema.org/draft/2020-12/json-schema-core.html>.

10.2. Informative References

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/rfc/rfc6749>.

[I-D.ietf-wimse-arch] Rosenzweig, Y. and H. Tschofenig, "Workload Identity in Multi System Environments (WIMSE) Architecture", Work in Progress, Internet-Draft, draft-ietf-wimse-arch, 2025, <https://datatracker.ietf.org/doc/draft-ietf-wimse-arch/>.

[LANDAUER] Landauer, R., "Irreversibility and Heat Generation in the Computing Process", IBM Journal of Research and Development 5(3):183-191, 1961, <https://doi.org/10.1147/rd.53.0183>.

[MCP-SPEC] Anthropic, "Model Context Protocol Specification", 2024, <https://modelcontextprotocol.io/specification>.

[W3C-DID] Sporny, M. and D. Longley, "Decentralized Identifiers (DIDs) v1.0", 2022, <https://www.w3.org/TR/did-core/>.

[OPENAPI] OpenAPI Initiative, "OpenAPI Specification v3.1.0", 2021, <https://spec.openapis.org/oas/v3.1.0>.

[AGENTCARD-SPEC] Tsoi, K. L., "AgentCard Reference Implementation and JSON Schema", 2026, <https://github.com/kwailapt/AgentCard>.

[AGENTCARD-MCP] Tsoi, K. L., "agentcard-mcp: MCP Server for AgentCard Identity", 2026, <https://github.com/kwailapt/AgentCard/tree/main/adapters/python/mcp-server>.

Acknowledgements

The author thanks the Aevum Network community for feedback on the PACR-derived metadata fields, and the MCP ecosystem for validating the dot-namespaced capability identifier convention through practical implementation.

The Landauer pricing floor concept was first articulated in the context of multi-agent economic systems in the Aevum Network specification (2026). Its inclusion in AgentCard reflects the broader principle that computational resources are physically bounded and should be declared honestly.

Author’s Address

Kwai Lap Tsoi
Aevum Network
Email: kwailapt@gmail.com
URI: https://github.com/kwailapt/AgentCard