Internet-Draft
Intended status: Standards Track
Expires: October 8, 2026

F. Batum

April 6, 2026

# AI Discovery and Retrieval Endpoint (AIDRE)
draft-batum-aidre-00

## Abstract

This document specifies the AI Discovery and Retrieval Endpoint (AIDRE), a protocol for publishing machine-oriented, canonical, and semantically retrievable content on the web. AIDRE defines a discovery document, collection metadata, retrieval interfaces, optional vector-native query support, and content representation rules for AI systems.

AIDRE aims to reduce redundant crawling, parsing, tokenization, and embedding of the same origin content while improving freshness, provenance, and interoperability for AI systems.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 8, 2026.



## Table of Contents

## 1. Introduction

Modern AI retrieval systems repeatedly process the same web content by crawling, extracting, chunking, and embedding text. This results in redundant computation, increased costs, and inconsistent interpretations.

AIDRE introduces a standardized mechanism for origins to expose canonical, machine-oriented retrieval interfaces, enabling direct semantic access to content.

## 2. Conventions and Terminology

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.

### 2.1. Terms

Origin: The web origin publishing AIDRE metadata.

Discovery Document: A JSON document describing endpoints and capabilities.

Collection: A logical grouping of content.

Chunk: A retrievable unit of canonical content.

Embedding Space: A defined vector representation space.

# 3. Discovery

## 3.1. Well-Known URI

An origin implementing AIDRE MUST expose the discovery document at

/.well-known/ai-discovery

over HTTPS using a well-known URI as defined in [RFC8615].

## 3.2. Media Type

The discovery document MUST be served as application/json.

## 3.3. DNS Considerations

Clients MUST NOT require DNS SRV lookup for discovery.

DNS HTTPS (SVCB) records MAY be used as optimization hints, as described in [RFC9460].

DNS SRV records are NOT RECOMMENDED for public interoperability.

## 3.4. Example

```
{
  "version": "1",
  "service": "AIDRE",
  "endpoints": {
    "search": "https://ai.example.com/search",
    "collections": "https://ai.example.com/collections",
    "chunk": "https://ai.example.com/chunks/{id}"
  },
  "capabilities": {
    "query_text": true,
    "query_vector": true
  }
}
```

# 4. Collections

Collections group retrievable content.

Each collection SHOULD define:

* name
* description
* visibility
* updated_at

## 5. Query Model

### 5.1. General Rule

A request MUST include exactly one of:

* query
* query_vector

### 5.2. Text Query

Servers MUST support text queries.

### 5.3. Vector Query

Servers SHOULD support vector queries.

If used, clients MUST specify embedding_space.

## 6. Search Endpoint

### 6.1. Method

Clients MUST use HTTP POST.

### 6.2. Request Example

```
{
  "query": "sso setup",
  "collection": "docs",
  "top_k": 5
}
```

### 6.3. Vector Example

```
{
  "query_vector": [0.01, -0.02],
  "embedding_space": "example-space",
  "collection": "docs"
}
```

## 7. Results

### 7.1. Structure

Responses MUST include a results array.

### 7.2. Example

```
{
  "results": [
    {
      "id": "doc_1#chunk_1",
      "score": 0.91,
      "metadata": {
        "updated_at": "2026-04-01T00:00:00Z",
        "canonical": true
      }
    }
  ]
}
```

### 7.3. Optional Fields

Servers MAY return:

* text
* semantic payload
* vectors

Servers SHOULD NOT return vectors by default.

## 8. Chunk Retrieval

Servers SHOULD support:

```
GET /chunks/{id}
```

## 9. Caching and Freshness

Servers SHOULD expose:

* content_hash
* updated_at
* etag

## 10. Error Handling

Errors MUST be JSON objects.

Example:

```
{
```

```
  "error": "unsupported_embedding_space",
  "message": "Embedding space not supported"
}
```

# 11. Security Considerations

AIDRE can make machine-oriented retrieval of content more efficient. That efficiency introduces risks, including accelerated scraping, exfiltration of structured content, abuse of vector-return paths, stale or malicious publication, and misleading provenance.

## 11.1. Retrieval Amplification

AIDRE can reduce the cost of large-scale content extraction for both legitimate and illegitimate clients. Deployments SHOULD consider rate limiting, authentication, authorization, abuse detection, and staged disclosure of higher-value representations.

## 11.2. Vector Disclosure and Leakage

Returning vectors can create additional disclosure risk beyond returning text alone. Depending on the embedding model, vectors MAY expose distributional or structural properties of source content and MAY enable downstream correlation, approximate membership inference, or other forms of analysis not intended by the publisher.

Accordingly:

* Servers SHOULD NOT return vectors by default.
* Servers SHOULD evaluate whether vector return is necessary for a given deployment.
* Sensitive or access-controlled collections SHOULD disable vector return unless there is a specific operational need.
* Deployments MAY use distinct policies for public and authenticated collections.

This specification does not require or assume that vector inversion is practical in all settings, but deployments SHOULD treat vector disclosure as a potentially sensitive operation.

## 11.3. Malicious or Stale Canonical Content

AIDRE gives publishers a machine-oriented canonical surface. If that surface is compromised, stale, poisoned, or maliciously altered, downstream systems MAY ingest or trust incorrect content at scale.

Deployments SHOULD therefore consider content review workflows, publication approvals, deprecation semantics, rollback procedures, and freshness validation.

## 11.4. Signed Provenance

Deployments MAY attach signed provenance metadata to responses, collections, or chunk resources. Signed provenance can help clients verify origin authenticity, content integrity, and publication scope.

If signed provenance is used, deployments SHOULD define:

* what is signed,
* which keys are authoritative,
* how keys are discovered,
* signature lifetime and rotation policy,
* the failure behavior when signature validation does not succeed.

A deployment using signed provenance SHOULD ensure that key discovery is bound to the publisher's trust model, for example via HTTPS on the publisher origin or another integrity-protected mechanism.

### 11.5. Query Abuse

Query inputs, including query vectors, can be adversarial. Servers SHOULD validate input size, dimensionality, and request shape, and SHOULD protect retrieval infrastructure against resource exhaustion.

### 11.6. Access-Controlled Collections

AIDRE does not imply that all collections are public. Private or tenant-scoped collections MUST be protected by appropriate authorization controls. Discovery documents SHOULD avoid revealing unnecessary detail about protected collections.

## 12. Privacy Considerations

Sensitive data SHOULD NOT be exposed unintentionally.

## 13. IANA Considerations

This document does not itself register a new media type.

However, this specification defines and uses the label application/aidre+json as the preferred representation for AIDRE request and response bodies. If AIDRE proceeds toward broader standardization and deployment, a future revision of this document or a companion specification SHOULD request IANA registration for that media type in accordance with the procedures applicable at that time.

Until such registration occurs, implementations SHOULD treat application/aidre+json as a provisional media type for experimental and pre-standard deployment.

## 14. Media Types

### 14.1. AIDRE JSON Media Type

AIDRE request and response bodies SHOULD use the media type application/aidre+json.

Clients MUST send a Content-Type header of application/aidre+json when sending AIDRE request bodies.

Clients SHOULD send an Accept header of application/aidre+json.

Servers MAY accept application/json for compatibility, but a conformant implementation SHOULD prefer application/aidre+json.

### 14.2. Character Encoding

AIDRE JSON payloads MUST use UTF-8 encoding.

## 15. HTTP Usage

### 15.1. Methods

The discovery and collection resources MUST support HTTP GET.

The search resource MUST support HTTP POST.

The chunk dereference resource MUST support HTTP GET.

### 15.2. Request Headers

Clients SHOULD send:

* Accept: application/aidre+json

Clients sending request bodies MUST send:

* Content-Type: application/aidre+json

### 15.3. Response Headers

Servers SHOULD include a Content-Type response header set to application/aidre+json for successful AIDRE responses.

Servers MAY include ETag, Last-Modified, Cache-Control, and RateLimit fields where appropriate.

### 15.4. Status Codes

Servers SHOULD use the following HTTP status codes consistently:

* 200 OK: successful request.
* 201 Created: resource created, if an extension defines creation.

*  400 Bad Request: syntactically invalid request.
*  401 Unauthorized: authentication required or failed.
*  403 Forbidden: authenticated but not permitted.
*  404 Not Found: resource identifier not found.
*  409 Conflict: request conflicts with resource state.
*  415 Unsupported Media Type: unsupported Content-Type.
*  422 Unprocessable Entity: semantically invalid request, such as unsupported embedding dimensionality.
*  429 Too Many Requests: rate limit exceeded.
*  500 Internal Server Error: unexpected server error.
*  503 Service Unavailable: temporary overload or maintenance.

# 16.  Request and Response Envelope

## 16.1.  General Response Shape

A successful search response MUST be a JSON object containing a results member whose value is an array.

A successful response SHOULD also contain:

*  request_id
*  collection
*  meta

## 16.2.  Response Envelope Example

```
{
  "request_id": "req_7f4c1c",
  "collection": "docs",
  "results": [
    {
      "id": "doc_123#chunk_7",
      "score": 0.92,
      "metadata": {
        "updated_at": "2026-04-01T10:00:00Z",
        "canonical": true,
        "visibility": "public"
      }
    }
  ],
  "meta": {
    "returned": 1,
    "top_k": 3
  }
}
```

## 16.3.  Request Identifiers

Servers SHOULD generate a request identifier for each request.

A request identifier SHOULD be unique within operationally relevant scope and SHOULD be suitable for debugging and audit correlation.

## 17. Error Model

### 17.1. Error Envelope

Errors MUST be JSON objects.

Error responses SHOULD contain:

* error
* message
* details
* request_id

### 17.2. Error Example

```
{
  "error": "unsupported_embedding_space",
  "message": "The requested embedding space is not supported.",
  "details": {
    "embedding_space": "example:unsupported-space"
  },
  "request_id": "req_7f4c1c"
}
```

### 17.3. Error Codes

Servers SHOULD use stable, machine-readable error values. Suggested values include:

* invalid_request
* unsupported_embedding_space
* invalid_embedding_dimension
* unsupported_return_field
* unauthorized
* forbidden
* not_found
* rate_limited
* internal_error

## 18. Pagination and Limits

### 18.1. top_k Semantics

The top_k member requests the maximum number of results returned.

If top_k is omitted, the server MUST apply a documented default.

Servers MUST reject non-positive top_k values.

Servers MAY enforce an implementation-defined maximum top_k.

If the client requests a top_k larger than the server maximum, the server MUST either reject the request with an error or clamp the value according to documented behavior.

### 18.2. Pagination

Search endpoints MAY support pagination.

If pagination is supported, the server SHOULD use an opaque cursor scheme.

The client MUST treat cursor values as opaque.

### 18.3. Pagination Envelope Example

```
{
  "request_id": "req_9aa1",
  "collection": "docs",
  "results": [],
  "meta": {
    "returned": 0,
    "top_k": 10,
    "next_cursor": "eyJvZmZzZXQiOjEwMH0"
  }
}
```

## 19. Embedding Compatibility Rules

### 19.1. Embedding Space Identification

Each embedding space advertised by the server MUST include:

* id
* dimensions
* distance

It SHOULD also include:

* normalized
* provider
* model
* revision

### 19.2. Dimensionality

If a query_vector length does not match the dimensions declared for the requested embedding space, the server MUST reject the request with 422 Unprocessable Entity.

### 19.3. Distance Function

The server MUST evaluate query vectors according to the distance function declared by the embedding space.

Clients MUST NOT assume cosine similarity unless the embedding space declaration says so.

### 19.4. Normalization

If an embedding space declares normalized=true, clients SHOULD send normalized vectors.

A server MAY normalize vectors on receipt, but such behavior SHOULD be documented.

### 19.5. Compatibility Failure Example

```
{
  "error": "invalid_embedding_dimension",
  "message": "The supplied query vector has length 1024, but the
    embedding space requires 3072.",
  "details": {
    "embedding_space":
      "openai:text-embedding-3-large:3072:cosine:v1",
    "expected_dimensions": 3072,
    "actual_dimensions": 1024
  }
}
```

## 20. Representation Negotiation

### 20.1. Return Object

The return object requests optional fields in result members.

Supported fields MAY include:

* ids
* metadata
* text
* semantic_payload
* vectors

### 20.2. Server Behavior

If a client requests an unsupported return field, the server MUST either:

* ignore the field, or
* reject the request with a machine-readable error.

The chosen behavior MUST be documented by the deployment.

### 20.3. Unknown Fields

Clients MUST ignore unknown response fields.

Servers SHOULD ignore unknown request fields unless doing so would create ambiguous or unsafe behavior.

## 21. Rate Limiting

### 21.1. General

Servers MAY apply rate limiting.

If rate limiting is applied, servers SHOULD expose limit state using HTTP rate limit fields defined in [RFC9333].

### 21.2. Suggested Fields

Deployments SHOULD consider exposing:

* RateLimit-Limit
* RateLimit-Remaining
* RateLimit-Reset

### 21.3. Related Specifications

Deployments are encouraged to align with existing HTTP rate limit field specifications rather than inventing deployment-specific headers when interoperable fields are sufficient.

## 22. Schema and OpenAPI Conformance

### 22.1. Companion Artifacts

This specification MAY be accompanied by JSON Schema, OpenAPI, or similar machine-readable descriptions of AIDRE messages and resources.

### 22.2. Normative Authority

Such artifacts are useful for tooling, validation, code generation, testing, and documentation. However, unless explicitly stated otherwise, this document remains the normative definition of the protocol.

If an OpenAPI description, JSON Schema, or other companion artifact conflicts with this document, this document takes precedence.

### 22.3. Conformance Guidance

Deployments SHOULD ensure that published companion artifacts are kept consistent with the protocol version advertised in the discovery document.

## 23. Versioning and Extensibility

### 23.1. Discovery Version

The discovery document MUST contain a version member.

### 23.2. Compatibility Rule

Backward-compatible additions SHOULD be made by adding new fields.

Clients MUST ignore unknown fields.

Backward-incompatible changes SHOULD be introduced through a new protocol version.

### 23.3. Extension Members

Deployments MAY define extension members.

Extension members SHOULD use names that minimize collision risk.

## 24. HTTP Exchange Examples

### 24.1. Discovery Request Example

Request:

```
GET /.well-known/ai-discovery HTTP/1.1
Host: example.com
Accept: application/aidre+json, application/json
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/aidre+json
Cache-Control: max-age=300
ETag: "disc-v1-9f2a"

{
  "version": "1",
  "service": "AIDRE",
  "organization": "Example Corp",
  "endpoints": {
    "collections": "https://ai.example.com/collections",
    "search": "https://ai.example.com/search",
    "chunk": "https://ai.example.com/chunks/{id}"
  },
```

```
  "capabilities": {
    "query_text": true,
    "query_vector": true,
    "return_text": true,
    "return_semantic_payload": true,
    "return_vectors": false,
    "delta_sync": false
  },
  "embedding_spaces": [
    {
      "id": "openai:text-embedding-3-large:3072:cosine:v1",
      "dimensions": 3072,
      "distance": "cosine",
      "normalized": true
    }
  ],
  "auth": {
    "type": "none"
  }
}
```

### 24.2. Search Request Example

Request:

```
POST /search HTTP/1.1
Host: ai.example.com
Accept: application/aidre+json
Content-Type: application/aidre+json

{
  "query_vector": [0.013, -0.028, 0.442],
  "embedding_space":
    "openai:text-embedding-3-large:3072:cosine:v1",
  "collection": "docs",
  "top_k": 3,
  "return": {
    "ids": true,
    "metadata": true,
    "text": false,
    "semantic_payload": true,
    "vectors": false
  }
}
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/aidre+json
RateLimit-Limit: 100
RateLimit-Remaining: 99
RateLimit-Reset: 60

{
  "request_id": "req_7f4c1c",
```

```
  "collection": "docs",
  "results": [
    {
      "id": "doc_123#chunk_7",
      "score": 0.92,
      "source": {
        "url": "https://example.com/docs/sso/setup",
        "title": "SSO Setup",
        "section": "Prerequisites"
      },
      "metadata": {
        "updated_at": "2026-04-01T10:00:00Z",
        "canonical": true,
        "visibility": "public",
        "content_hash": "sha256:abcd..."
      },
      "semantic_payload": {
        "type": "application/semantic+json",
        "claims": [
          {
            "subject": "SAML SSO",
            "predicate": "requires",
            "object": "domain verification"
          }
        ]
      }
    }
  ],
  "meta": {
    "returned": 1,
    "top_k": 3
  }
}
```

### 24.3. Compatibility Error Example

Request:

```
POST /search HTTP/1.1
Host: ai.example.com
Accept: application/aidre+json
Content-Type: application/aidre+json

{
  "query_vector": [0.1, 0.2],
  "embedding_space":
    "openai:text-embedding-3-large:3072:cosine:v1",
  "collection": "docs"
}
```

Response:

```
HTTP/1.1 422 Unprocessable Entity
Content-Type: application/aidre+json
```

```
{
  "error": "invalid_embedding_dimension",
  "message": "The supplied query vector has length 2, but the
    embedding space requires 3072.",
  "details": {
    "embedding_space":
      "openai:text-embedding-3-large:3072:cosine:v1",
    "expected_dimensions": 3072,
    "actual_dimensions": 2
  },
  "request_id": "req_err_11"
}
```

# 25. ABNF Summary

The following ABNF, using the notation from [RFC5234], summarizes the principal AIDRE JSON member names. It is descriptive of member names and presence expectations and is not a complete JSON grammar.

```
query-member           = %s"query"
query-vector-member    = %s"query_vector"
embedding-space-member = %s"embedding_space"
collection-member      = %s"collection"
topk-member            = %s"top_k"
return-member          = %s"return"
results-member         = %s"results"
metadata-member        = %s"metadata"
request-id-member      = %s"request_id"
next-cursor-member     = %s"next_cursor"
```

# 26. References

## 26.1. Normative References


[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", RFC 8174.

[RFC5234] Crocker, D., and P. Overell, "Augmented BNF for Syntax Specifications", RFC 5234.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615.


## 26.2. Informative References


[RFC9460] Schwartz, B., "Service Binding and Parameter Specification via the DNS", RFC 9460.

[RFC9333] Polli, R. and M. Martinez, "The RateLimit Fields for HTTP", RFC 9333.

## 27. Additional Design Considerations

### 27.1. Why Text Query Remains Mandatory

Text query support remains mandatory for baseline interoperability. Not all clients and deployments will share an embedding space, and a text path allows a client to interact with an AIDRE deployment even when vector compatibility has not been negotiated.

### 27.2. Why Vector Return Is Optional

Vector-native querying and vector disclosure are distinct concerns. A deployment may wish to support vector-native query semantics while refusing to disclose stored vectors. This separation allows the protocol to reduce redundant text processing without forcing a single disclosure model.

## Author’s Address

Fatih Batum
Istanbul, Turkiye
Email: fatih@batum.gen.tr