
Workgroup: Network Working Group
Internet-Draft: draft-payment-discovery-00
Published: 18 May 2026
Intended Status: Informational
Expires: 19 November 2026
Authors: B. Ryan J. Moxey R. Sproule S. Ragsdale
Tempo Labs Tempo Labs Merit Systems Merit Systems

Service Discovery for HTTP Payment Authentication

Abstract

This document defines a service discovery framework for the "Payment" HTTP authentication scheme. Services publish an OpenAPI document annotated with payment extensions that describe pricing, payment methods, and intent types. The OpenAPI document serves as the canonical machine-readable contract, providing both payment metadata and input schemas so that agents can discover and invoke endpoints. The runtime 402 challenge remains authoritative for all payment parameters.

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 19 November 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

This document may not be modified, and derivative works of it may not be created, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
2. Requirements Language	4
3. Terminology	4
4. OpenAPI Discovery	4
4.1. Document Location	4
4.2. Required Top-Level Fields	4
4.3. Service Extension: x-service-info	4
4.3.1. Categories	5
4.3.2. Documentation Links	5
4.4. Payment Extension: x-payment-info	5
4.4.1. Payment Offer Object	6
4.4.2. Payment Offer Examples	6
4.5. 402 Response Declaration	9
4.6. Input Schema	9
4.7. Caching	9
4.8. Example OpenAPI Document	9
5. Relationship to the 402 Challenge	12
6. Security Considerations	12
6.1. Discovery Spoofing	12
6.2. Information Disclosure	12
6.3. Cross-Origin Requests	12
7. IANA Considerations	12
8. References	12
8.1. Normative References	12
8.2. Informative References	13

Appendix A. Registry and Aggregator Guidance	14
A.1. Registries	14
A.2. Aggregators	14
Appendix B. Comparison with Prior Art	14
B.1. CoRE Resource Directory (RFC 9176)	14
B.2. Agent2Agent Protocol (A2A)	14
B.3. MCP Registry	15
B.4. x402 Protocol	15
B.5. OpenAPI-First Discovery (x402scan)	15
B.6. ERC-8004 (Trustless Agents)	15
Appendix C. JSON Schema for x-payment-info	15
Appendix D. JSON Schema for x-service-info	16
Acknowledgments	17
Authors' Addresses	17

1. Introduction

The "Payment" HTTP authentication scheme [[I-D.httpauth-payment](#)] enables servers to require payment for resource access using the HTTP 402 status code. While the 402 challenge provides all information needed to complete a single paid exchange, clients and agents benefit from discovering payment-enabled services before initiating requests.

This specification defines a discovery mechanism based on OpenAPI [[OPENAPI](#)]. Services publish an OpenAPI document annotated with two extensions:

- `x-service-info`: Top-level service metadata including categories and documentation links.
- `x-payment-info`: Per-operation payment requirements including one or more payment offers.

OpenAPI provides both payment metadata and input schemas, enabling agents to discover and invoke endpoints without additional documentation.

Discovery is **OPTIONAL**. Servers **MAY** implement this mechanism to improve client experience. Clients **MUST NOT** require discovery to function; the 402 challenge in [[I-D.httpauth-payment](#)] is always authoritative.

2. 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.

3. Terminology

Service An HTTP origin that accepts payment via the "Payment" authentication scheme.

Payable Operation An API operation that requires payment, indicated by a 402 response and x-payment-info extension in the OpenAPI document.

Payment Offer A discovered payment alternative for a payable operation. Multiple payment offers correspond to alternative runtime Payment challenges for the same operation.

4. OpenAPI Discovery

Services that support discovery **MUST** publish an OpenAPI 3.x [OPENAPI] document that describes their API surface, including payment-enabled operations.

4.1. Document Location

The OpenAPI document **MUST** be accessible at:

```
GET /openapi.json
```

The document **MUST** be served over HTTPS with Content-Type: application/json.

4.2. Required Top-Level Fields

The OpenAPI document **MUST** include the following standard fields:

- `openapi`: The OpenAPI version (e.g., "3.1.0").
- `info.title`: The service name.
- `info.version`: The API version.
- `paths`: At least one path with operations.

4.3. Service Extension: x-service-info

The OpenAPI document **MAY** include a top-level x-service-info extension object to provide service metadata that is not part of the standard OpenAPI specification.

Field	Type	Required	Description
categories	array	OPTIONAL	Service categories (see Section 4.3.1).
docs	object	OPTIONAL	Documentation and reference links (see Section 4.3.2).

Table 1

4.3.1. Categories

The `categories` field, when present, **MUST** be an array of strings. Category values are free-form; services **MAY** use any string value. The following values are **RECOMMENDED** as a starting vocabulary:

```
communication, compute, data, developer-tools,
media, search, social, storage, travel
```

Category values **SHOULD** be lowercase, use hyphens for multi-word values, and be concise. Registries **SHOULD** limit services to no more than 5 categories. Clients **SHOULD** ignore category values they do not recognize.

4.3.2. Documentation Links

The `docs` field, when present, **MUST** be a JSON object with the following optional fields:

Field	Type	Description
apiReference	string (URI)	API reference documentation URL.
homepage	string (URI)	Main documentation or landing page.
llms	string (URI)	LLM-friendly documentation URL (see [LLMS-TXT]).

Table 2

All URI values **MUST** conform to [\[RFC3986\]](#).

4.4. Payment Extension: x-payment-info

Each payable operation **MUST** include the `x-payment-info` extension object on the operation. This extension describes one or more payment offers for the operation.

The extension supports two equivalent forms:

- Single-offer shorthand: an object containing the fields of a single payment offer directly.
- Multi-offer form: an object containing an `offers` array of one or more payment offer objects.

Servers publishing new discovery documents **SHOULD** use the multi-offer form. Clients and registries **MUST** accept both forms. When the single-offer shorthand is used, clients and registries **MUST** treat it as equivalent to a multi-offer form containing exactly one payment offer.

When multiple offers are present, clients **SHOULD** treat them as alternative ways to access the same operation. At runtime, the client selects one offer and fulfills the corresponding 402 challenge.

4.4.1. Payment Offer Object

Field	Type	Required	Description
intent	string	REQUIRED	"charge" (per-request) or "session" (pay-as-you-go).
method	string	REQUIRED	Payment method identifier (e.g., "tempo", "stripe").
amount	string or null	REQUIRED	Cost in base currency units. null indicates dynamic pricing.
currency	string	OPTIONAL	Currency identifier. For blockchain methods: token contract address. For fiat: ISO 4217 code.
description	string	OPTIONAL	Human-readable pricing note.

Table 3

The amount field is **REQUIRED** but its value **MAY** be null to support offers where pricing depends on request parameters (e.g., variable-cost operations). When non-null, the value **MUST** be a string of ASCII digits (0-9) representing a non-negative integer in the smallest denomination of the currency (e.g., cents for USD, wei for ETH). Leading zeros **MUST NOT** be used except for the value "0". This format is consistent with the amount field defined in the request object of [I-D.httpauth-payment].

4.4.2. Payment Offer Examples

The following examples illustrate common multi-offer patterns.

4.4.2.1. Same Intent, Different Currency

The same charge intent can be offered in multiple currencies for the same operation:

```
{
  "x-payment-info": {
    "offers": [
      {
        "intent": "charge",
        "method": "tempo",
        "amount": "500",
        "currency":
          "0x20c0000000000000000000000000000000000000000000000000000000000000"
      },
      {
        "intent": "charge",
        "method": "tempo",
        "amount": "500",
        "currency":
          "0x20c0000000000000000000000000000000000000000000000000000000000000b9537d11c60e8b50"
      }
    ]
  }
}
```

4.4.2.2. Multiple Methods Under One Intent

The same charge intent can be offered through different payment methods:

```
{
  "x-payment-info": {
    "offers": [
      {
        "intent": "charge",
        "method": "tempo",
        "amount": "500",
        "currency":
          "0x20c0000000000000000000000000000000000000000000000000000000000000b9537d11c60e8b50"
      },
      {
        "intent": "charge",
        "method": "stripe",
        "amount": "5",
        "currency": "usd"
      }
    ]
  }
}
```

4.4.2.3. Fixed and Dynamic Offers

An operation can mix fixed-price and dynamic-price offers in the same intent set. In this example, Tempo is fixed-price while Stripe is dynamic:

```
{
  "x-payment-info": {
    "offers": [
      {
        "intent": "charge",
        "method": "tempo",
        "amount": "500",
        "currency":
          "0x20c00000000000000000000000000000b9537d11c60e8b50"
      },
      {
        "intent": "charge",
        "method": "stripe",
        "amount": null,
        "currency": "usd",
        "description":
          "Price varies by processor fees and request size."
      }
    ]
  }
}
```

4.4.2.4. Multiple Methods and Intents

An operation can advertise multiple methods and multiple intents at the same time:

```
{
  "x-payment-info": {
    "offers": [
      {
        "intent": "session",
        "method": "tempo",
        "amount": "500",
        "currency":
          "0x20c0000000000000000000000000000000000000000000000000000000000000"
      },
      {
        "intent": "charge",
        "method": "tempo",
        "amount": "750",
        "currency":
          "0x20c00000000000000000000000000000b9537d11c60e8b50"
      },
      {
        "intent": "charge",
        "method": "stripe",
        "amount": "8",
        "currency": "usd"
      }
    ]
  }
}
```

4.5. 402 Response Declaration

Each payable operation **MUST** include a 402 response in its `responses` object:

```
responses:
  "402":
    description: "Payment Required"
```

This signals to clients that the operation may return a 402 challenge requiring payment.

4.6. Input Schema

Each operation **SHOULD** define its input schema using the standard OpenAPI `requestBody` field:

```
requestBody:
  content:
    application/json:
      schema:
        type: object
        properties:
          prompt:
            type: string
        required:
          - prompt
```

Input schemas enable agents to construct valid requests without additional documentation. Operations that omit input schemas **MAY** be marked as "schema-missing" by discovery clients and registries.

4.7. Caching

Servers **SHOULD** include `Cache-Control` headers. A maximum age of 5 minutes is **RECOMMENDED** for services whose capabilities change infrequently:

```
Cache-Control: max-age=300
```

Clients **SHOULD** respect cache headers and refetch when capabilities may have changed.

4.8. Example OpenAPI Document

```
{
  "openapi": "3.1.0",
  "info": {
    "title": "Example AI API",
    "version": "1.0.0"
  },
}
```

```
"x-service-info": {
  "categories": ["compute"],
  "docs": {
    "homepage": "https://api.example.com/docs",
    "llms": "https://api.example.com/llms.txt",
    "apiReference":
      "https://api.example.com/reference"
  }
},
"paths": {
  "/v1/chat/completions": {
    "post": {
      "summary": "Chat completions",
      "x-payment-info": {
        "offers": [
          {
            "intent": "charge",
            "method": "tempo",
            "amount": "500",
            "currency":
              "0x20c000000000000000000000000000000000000000"
          },
          {
            "intent": "charge",
            "method": "tempo",
            "amount": "500",
            "currency":
              "0x20c000000000000000000000000000b9537d11c60e8b50",
            "description":
              "Alternative Tempo asset for the same route."
          }
        ]
      }
    }
  },
  "requestBody": {
    "content": {
      "application/json": {
        "schema": {
          "type": "object",
          "properties": {
            "model": { "type": "string" },
            "messages": {
              "type": "array",
              "items": {
                "type": "object",
                "properties": {
                  "role": {
                    "type": "string"
                  },
                  "content": {
                    "type": "string"
                  }
                }
              },
              "required": ["role",
                "content"]
            }
          }
        },
        "required": ["model", "messages"]
      }
    }
  }
}
```

```
    }
  }
},
"responses": {
  "200": {
    "description": "Successful response"
  },
  "402": {
    "description": "Payment Required"
  }
}
},
"/v1/embeddings": {
  "post": {
    "summary": "Text embeddings",
    "x-payment-info": {
      "intent": "charge",
      "method": "tempo",
      "amount": null,
      "currency":
        "0x20c0000000000000000000000000000000000000000000000000000000000000",
      "description": "Price varies by model
        and token count."
    },
    "requestBody": {
      "content": {
        "application/json": {
          "schema": {
            "type": "object",
            "properties": {
              "model": { "type": "string" },
              "input": { "type": "string" }
            },
            "required": ["model", "input"]
          }
        }
      }
    },
    "responses": {
      "200": {
        "description": "Successful response"
      },
      "402": {
        "description": "Payment Required"
      }
    }
  }
}
}
```

5. Relationship to the 402 Challenge

Discovery metadata is advisory. The 402 challenge defined in [I-D.httpauth-payment] is always authoritative.

Specifically:

- If discovery indicates payment offer details that differ from the 402 challenge, including method, intent, amount, or currency, the 402 challenge takes precedence.
- Clients **MUST NOT** cache discovery data as a substitute for processing 402 challenges.

Discovery exists to help clients and agents find and evaluate services before making requests, not to replace the runtime payment negotiation defined by the core protocol.

6. Security Considerations

6.1. Discovery Spoofing

Discovery information is not cryptographically authenticated beyond HTTPS transport security. Clients **MUST NOT** rely on discovery metadata for security decisions. The 402 challenge is authoritative for all payment parameters.

6.2. Information Disclosure

OpenAPI documents reveal payment capabilities, endpoint structure, input schemas, and pricing to unauthenticated clients. Service operators **SHOULD** consider whether this disclosure is acceptable for their use case.

6.3. Cross-Origin Requests

Browser-based clients may need to access discovery endpoints cross-origin. Servers that intend to support browser-based clients **SHOULD** include appropriate CORS headers on OpenAPI document responses.

7. IANA Considerations

This document has no IANA actions.

8. References

8.1. 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/info/rfc2119>>.

[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/info/rfc3986>>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/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/info/rfc8259>>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <<https://www.rfc-editor.org/info/rfc9110>>.

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

[I-D.httpauth-payment] Ryan, B., "The 'Payment' HTTP Authentication Scheme", January 2026, <<https://datatracker.ietf.org/doc/draft-ryan-httpauth-payment/>>.

8.2. Informative References

[RFC9176] Amsüss, C., Ed., Shelby, Z., Koster, M., Bormann, C., and P. van der Stok, "Constrained RESTful Environments (CoRE) Resource Directory", RFC 9176, DOI 10.17487/RFC9176, April 2022, <<https://www.rfc-editor.org/info/rfc9176>>.

[A2A] Google, "Agent2Agent Protocol Specification", 2025, <<https://github.com/a2aproject/A2A>>.

[MCP-REGISTRY] Anthropic, "Model Context Protocol Registry", 2025, <<https://github.com/modelcontextprotocol/registry>>.

[X402] Coinbase, "x402: HTTP Payment Protocol", 2025, <<https://github.com/coinbase/x402>>.

[ERC-8004] "ERC-8004: Trustless Agents Registry", 2025, <<https://eips.ethereum.org/EIPS/eip-8004>>.

[LLMS-TXT] "llms.txt - A Proposal to Standardise LLM-Friendly Documentation", 2024, <<https://llmstxt.org/>>.

Appendix A. Registry and Aggregator Guidance

This appendix provides informative guidance for building registries and aggregators on top of the discovery mechanism defined in this specification.

A.1. Registries

A registry is a server that discovers, validates, and indexes payment-enabled services into a searchable catalog. Registries **MAY** discover services by:

- Crawling OpenAPI documents from submitted domains.
- Accepting domain submissions from service operators.
- Consuming snapshots from other registries.

If a domain serves a valid OpenAPI document with `x-payment-info` extensions over HTTPS, that constitutes sufficient proof of domain ownership.

Registries **SHOULD** re-crawl services periodically (at least every 24 hours is **RECOMMENDED**). If the discovery document becomes invalid or unreachable, the registry **SHOULD** delist the service after 7 or more consecutive failures.

Registries **SHOULD** enforce crawl constraints: HTTPS only, 10-second timeouts, 64 KB size limits, and rate limiting.

A.2. Aggregators

Aggregators consume registry data and layer on their own views: curating (filtering by quality or vertical), enriching (adding trust scores, uptime, volume data), reshaping (exposing agent-native formats such as `llms.txt` [LLMS-TXT]), or federating (merging data from multiple registries).

Aggregators are not required to use the registry API schema. The only universal contract is the OpenAPI discovery mechanism defined in [Section 4](#).

Appendix B. Comparison with Prior Art

B.1. CoRE Resource Directory (RFC 9176)

The CoRE Resource Directory [RFC9176] defines push registration with leased lifetimes for constrained IoT devices. This specification uses crawl-based registration, which better suits HTTP services.

B.2. Agent2Agent Protocol (A2A)

The A2A Protocol [A2A] uses `/well-known/agent-card.json` as a self-describing service endpoint. This specification uses OpenAPI as the discovery mechanism, providing richer schema information.

B.3. MCP Registry

The MCP Registry [[MCP-REGISTRY](#)] implements a three-layer architecture with reverse-DNS namespacing and SHA-256 package integrity. This specification uses domain authority rather than OAuth-based registration.

B.4. x402 Protocol

The x402 protocol [[X402](#)] uses HTTP 402 responses as the primary payment signal. This specification separates discovery (pre-request) from the payment challenge (at-request).

B.5. OpenAPI-First Discovery (x402scan)

The x402scan project uses OpenAPI documents as the canonical discovery signal, with /.well-known/x402 as a fallback. This specification adopts the same OpenAPI-first approach, using x-payment-info as the payment extension with fields consistent with the Payment authentication scheme.

B.6. ERC-8004 (Trustless Agents)

ERC-8004 [[ERC-8004](#)] defines on-chain identity registries and domain verification. This specification operates entirely off-chain but is compatible with future on-chain anchoring.

Appendix C. JSON Schema for x-payment-info

The following JSON Schema defines the structure of the x-payment-info OpenAPI extension. Tooling authors **SHOULD** validate payment extensions against this schema.

```
{
  "$schema":
    "https://json-schema.org/draft/2020-12/schema",
  "title": "x-payment-info",
  "oneOf": [
    { "$ref": "#/$defs/offer" },
    {
      "type": "object",
      "required": ["offers"],
      "properties": {
        "offers": {
          "type": "array",
          "minItems": 1,
          "items": { "$ref": "#/$defs/offer" }
        }
      },
      "additionalProperties": false
    }
  ],
  "$defs": {
    "offer": {
      "type": "object",
```

```
"required": ["intent", "method", "amount"],
"properties": {
  "intent": {
    "type": "string",
    "enum": ["charge", "session"]
  },
  "method": {
    "type": "string"
  },
  "amount": {
    "oneOf": [
      { "type": "null" },
      {
        "type": "string",
        "pattern": "^(0|[1-9][0-9]*)$"
      }
    ]
  },
  "currency": {
    "type": "string"
  },
  "description": {
    "type": "string"
  }
},
"additionalProperties": false
}
}
```

Appendix D. JSON Schema for x-service-info

The following JSON Schema defines the structure of the x-service-info OpenAPI extension.

```
{
  "$schema":
    "https://json-schema.org/draft/2020-12/schema",
  "title": "x-service-info",
  "type": "object",
  "properties": {
    "categories": {
      "type": "array",
      "items": { "type": "string" }
    },
    "docs": {
      "type": "object",
      "properties": {
        "apiReference": {
          "type": "string",
          "format": "uri"
        },
        "homepage": {
          "type": "string",
          "format": "uri"
        },
        "llms": {
          "type": "string",
          "format": "uri"
        }
      }
    }
  }
}
```

Acknowledgments

The authors thank the contributors to the MPP Registry reference implementation and the x402scan project, whose operational experience informed this specification.

Authors' Addresses

Brendan Ryan

Tempo Labs

Email: brendan@tempo.xyz

Jake Moxey

Tempo Labs

Email: jake@tempo.xyz

Ryan Sproule

Merit Systems

Email: ryan@merit.systems

Sam Ragsdale

Merit Systems

Email: sam@merit.systems