

# AI Agent Discovery and Invocation Protocol

**AIDIP**

*draft-cui-ai-agent-discovery-invocation-01*

Presenter: **Yihan Chao** Beijing Zhongguancun Laboratory

Co-authors:

Yong Cui (Tsinghua Univ.) · Chenguang Du (ZGC Lab)

**IETF 125 · March 2026**

# Motivation: Why Do We Need AIDIP?

AI agents are everywhere — translation, summarization, code generation. But every framework has its own API. Integrating agents across platforms means writing custom code for each one. There is no standard way to discover or invoke agents.

## ① Platform Silos

LangChain, AutoGen, custom deployments — each has its own agent description format and invocation API. They cannot interoperate.

## ② High Integration Cost

Building a multi-agent system today requires repeated custom glue code for every framework. This doesn't scale.

## ③ No Interoperability

Cross-platform, cross-organization agent collaboration is practically impossible without a shared standard.

→ AIDIP fills this gap with a unified standard for agent description, discovery, and invocation.

# AIDIP: Four Core Components

+ ASR (NEW in v01)

## ① Agent Metadata Spec

Standard JSON Schema describing any agent's capabilities, endpoint, authentication, and I/O format. Think of it as a machine-readable business card for agents.

## ② Discovery Mechanism

Agents register with a registry. Clients find them via structured attribute filters — or natural-language semantic search. Both modes can be combined in one request.

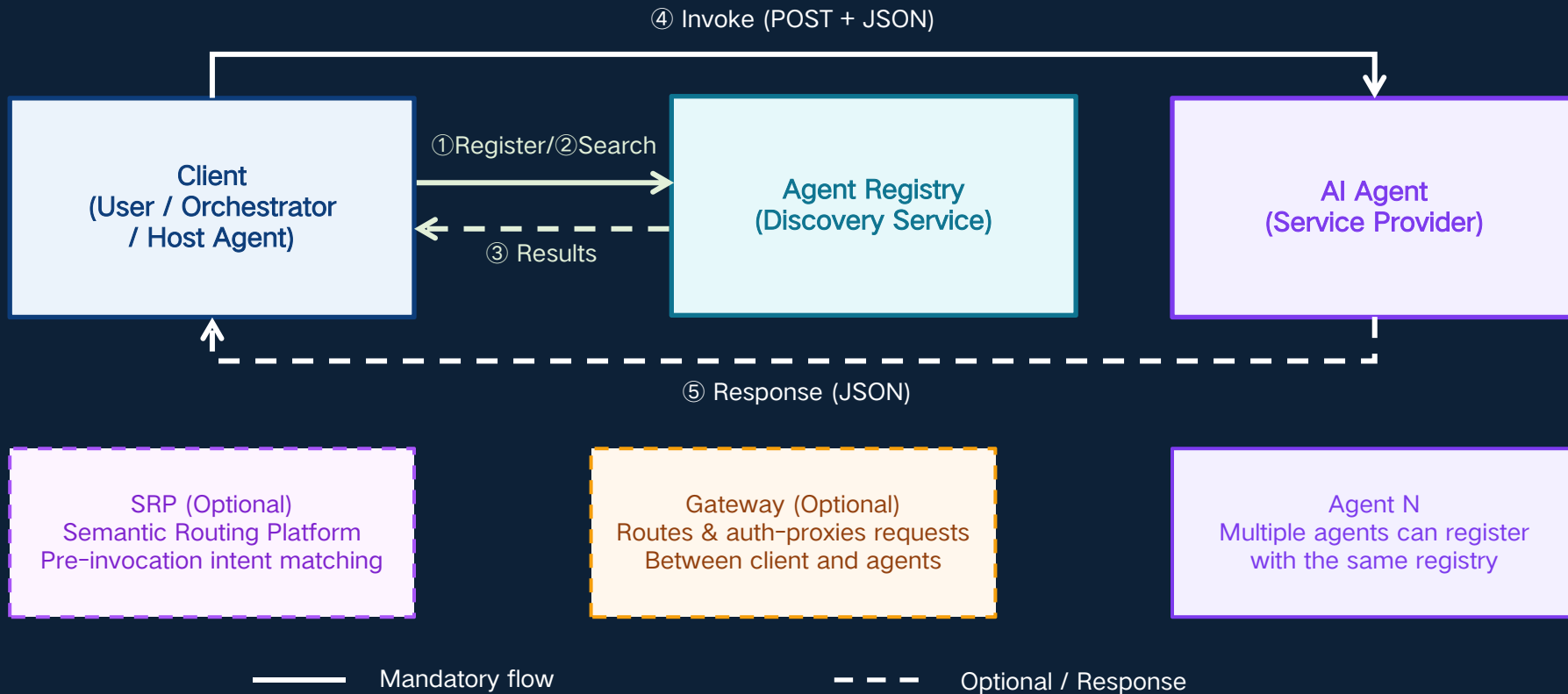
## ③ Invocation Interface

HTTP POST + JSON. Same pattern for every agent, every platform. Standard status codes: 200 / 4xx / 5xx. Error responses follow a uniform format. No custom protocol.

## ④ Security Framework

TLS mandatory for all traffic. OAuth 2.0, API keys, and mutual TLS for authentication. Rate limiting and per-client access control required by spec.

# System Architecture



# Agent Metadata + Discovery Mechanism

## Metadata Fields (required for every agent)

<b>id</b>	Globally unique UUID
<b>name</b>	Human-readable name
<b>capabilities</b>	e.g. translation, summarization
<b>tags</b>	Fine-grained categories
<b>endpoint</b>	Invocation URL (HTTPS)
<b>authentication</b>	api_key / oauth2_bearer / mTLS
<b>operations</b>	Input + output JSON Schema
<b>status</b>	available / busy / offline

## ① Attribute Filter

POST /agents/search with required capabilities, tags, languages.

Registry applies AND logic → returns ranked matches. Fast and deterministic.

```
{ "capabilities": ["translation"],  
  "filter": { "src_lang": "en", "tgt_lang":  
             "zh" } }
```

## ② Semantic NL Search

Client describes need in plain English. Registry uses embeddings or LLM to match agents — returns similarity scores.

Optional but backward-compatible: registries that don't support it simply ignore the query field.

```
{ "query": "summarize English legal docs into  
Chinese" }
```

# Agent Semantic Resolution (ASR) — NEW in v01

v01 NEW

( Intent + Context + Policy ) → Most Suitable Agent Endpoint(s) + Invocation Metadata

## What ASR answers:

NOT "where does named agent X live?" — that's DNS.  
BUT: "which agent is BEST SUITED for this specific task right now?"

Implemented by the Semantic Routing Platform (SRP) — sits before the standard discovery flow. SRP returns candidates; then normal invocation proceeds.

## Non-goals of ASR:

- ✗ Not a name→address resolver
- ✗ Does not replace DNS / URI registries
- ✗ Does not assign global persistent IDs
- ✗ SRP does not execute tasks

**Fully OPTIONAL.** Clients that don't use ASR continue with attribute-based discovery — zero changes needed.

## Backward compatible.

All mechanisms from v00 remain valid. ASR is strictly additive — it only adds, never removes.

# Example: Summarize English Legal Doc → Chinese

## ① Search

```
POST
/agents/search
{
  "capabilities":
  ["summarization"],
  "query":
  "summarize English
  legal doc to
  Chinese"
}
```

## ② Select

Review returned candidates — each with id, name, description, relevance score.

GET /agents/{id}  
→ fetch full metadata + exact input schema.

## ③ Invoke

```
POST
https://agent.example/summarize
{
  "content": "...",
  "output_language":
  "zh",
  "max_length":
  500
}
```

## ④ Response

```
HTTP 200 OK
{
  "summary":
  "...Chinese
  summary...",
  "language": "zh",
  "tokens_used":
  312
}
```

*Same 4-step flow for ANY AIDIP-compliant agent on ANY platform.*

# Summary

## ① Unified Agent Metadata

JSON Schema standard for capabilities, I/O format, authentication, and versioning.

## ② Two-Mode Discovery

Attribute filter + optional semantic NL search — single POST /agents/search endpoint.

## ③ Uniform Invocation Interface

HTTP POST + JSON. Standard status codes and error objects. Platform-agnostic.

## ④ Agent Semantic Resolution

Optional ASR via SRP — intent-based pre-selection. Fully backward-compatible.

## ⑤ Security Framework

TLS mandatory · OAuth 2.0 / API Key / mTLS · Rate limiting · Trust model extends to ASR.