

DMSC Working Group
Internet-Draft
Intended status: Standards Track
Expires: 13 August 2026

S. Sun
ICT, CAS
X. Zhang
CNIC, CAS
February 2026

Intent-based Agent Interconnection Protocol at Agent Gateway
draft-sz-dmsc-iaip-00

Abstract

This document specifies the Intent-based Agent Interconnection Protocol (IAIP) operating at the Agent Gateways (AG), which defines the interaction mechanisms between AI Agents (at the Agent Domain) and the AG (at the Interconnection Services Domain). This specification focuses on dynamic interconnection among agents based on semantic intent, rather than static network addressing alone. This protocols defines the mechanisms for agent registration via capability advertisement, Gateway Validation, Intent Resolution and Matching, Routing Decisions and Forwarding, enabling the discovery, selection and dispatching of intent queries based on agent capabilities and task requirements at AG.

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 5 August 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.

Table of Contents

1. Introduction	3
2. Conventions	3
3. Terminology	3
4. Problem Statement and Use Cases	4
4.1. Limitations of Address-Based Static Interconnection	4
4.2. Representative Use Case	5
5. Function Components	6
6. Protocol Overview	6
7. Message Formats	10
8. Protocol Operation and Core Procedures	12
8.1. Illustrative Protocol Examples	12
8.1.1. Example A: Agent Registration (CAP_ADV)	12
8.1.2. Example B: Intent Resolution Flow	12
8.1.3. Example C: Routing Feedback	13
8.2. Agent Access Component (AAC) Operations	13
8.2.1. Session Authentication and Integrity	13
8.2.2. Ingress Validation	14
8.3. Local Capability Registry (LCR) Management	14
8.3.1. Profile Creation and Updates	14
8.3.2. Lifecycle Maintenance	14
8.4. Intent Forwarding Routing Table (IFR) Decision Logic	14
8.4.1. Candidate Retrieval	14
8.4.2. Constraint-Based Filtering	15
8.4.3. Semantic Evaluation and Ranking	15
8.4.4. Selection and Fallback	15
8.5. Forwarding and Loop Prevention	15
8.6. Reliability and Error Handling	15
8.7. Protocol Processing procedure at Agent Gateways	16
9. Conclusions	18
10. Security Considerations	18
11. IANA Considerations	18
12. Normative References	18
Authors' Addresses	18

1. Introduction

The emergence of agentic networks marks a fundamental shift from static, address-based networking to semantic, intent-driven interactions. Autonomous agents powered by advanced artificial intelligence models are capable of reasoning, planning, and executing tasks on behalf of users or other agents. Interactions are increasingly expressed in terms of high-level objectives or natural language intents, rather than predefined service endpoints or static interfaces.

Traditional networking and service discovery mechanisms assume that communication targets are identified by fixed addresses, names, or service identifiers. These assumptions no longer hold in agentic environments, where the appropriate execution entity for a request may depend on semantic interpretation, contextual constraints, and dynamic system conditions. As a result, routing decisions based solely on static addressing or preconfigured bindings are insufficient to support flexible and scalable agent collaboration.

Intent-based interconnection addresses this gap by enabling the Agent Gateway to resolve and forward requests according to their semantic intent, rather than their destination address. By decoupling request dispatch from rigid topology and static identifiers, intent-based interconnection allows autonomous agents to be dynamically discovered, selected, and invoked based on their advertised capabilities.

This document introduces the Intent-based Agent Interconnection Protocol (IAIP) to provide a standardized mechanism for capability-based registration and validation, intent-aware discovery and routing at AG. IAIP is designed to operate as an application-layer protocol for interconnection service domain, complementing existing transport and networking infrastructure without requiring changes to underlying transport protocols.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119](#).

3. Terminology

AI Agent An agent is a software or hardware entity with autonomous

decision-making and execution capabilities, capable of perceiving the environment, acquiring contextual information, reasoning, and learning.

Agent Gateway (AG) A Functional Component in the Interconnection Service Domain that possesses the capabilities of agent registration, capability authentication, intent resolution and establishing interconnection between agents.

Intent A declarative expression of a desired outcome. In the intent-driven routing protocol, intent is represented at three conceptual layers:

- * ***Human Intent:** A high-level, possibly ambiguous expression originating from a human user or application (e.g., natural language input). Human Intent is out of scope for the protocol.
- * ***Task Intent:** An abstract, task-oriented description that captures the objective of a request, independent of any specific agent, algorithm, or execution plan.
- * ***Intent Descriptor:** A structured, machine-interpretable representation of Task Intent used within IAIP for routing and dispatching decisions.

Leader Agent (LA) Agent who issues tasks and launches interactions. There should only be one Leader Agent in a complete task execution process.

Partner Agent (PA) Agent who accepts tasks and provides services. After Partner Agent receives a task from the Leader Agent, it executes and returns the execution result.

4. Problem Statement and Use Cases

4.1. Limitations of Address-Based Static Interconnection

Existing network interconnection and gateway mechanisms are predominantly organized around static identifiers, such as network addresses, domain names, or service labels. At the application and service-dispatch layer, these mechanisms assume that the functionality associated with an agent behind a gateway is stable, explicitly addressable and functionally static. While effective for traditional client-server and microservice architectures, this assumption becomes increasingly inadequate for the dynamic interconnection required by agentic networks.

First, traditional interconnection protocols provide limited support for expressing semantic intent. High-level requests such as "analyze this dataset" or "resolve a billing issue" do not naturally correspond to a single predefined endpoint. Encoding semantic meaning into static addresses or service names requires manual configuration and tight coupling between task requesters and service providers, which undermines flexibility and scalability.

Second, static interconnection lacks adaptability to dynamic lifecycle of AI Agents. In the Agent Domain, agents may be instantiated, fine-tuned, or deprecated rapidly based on computational load or task requirements. Static bindings or name-based resolution mechanisms are generally unable to reflect real-time changes in agent's availability or intent-processing capacity, leading to service selection or routing dispatch failures.

Third, existing protocols offer limited support for intent-aware resolution. When a request arrives at the gateway without a specific destination address, traditional gateway have no mechanism to parse the intent and match it against a local registry of vector-based capabilities. This behavior is misaligned with agentic workflows, where ambiguous or underspecified intents are common and may require clarification, delegation, or escalation to more generalist agents.

These limitations motivate the need for an intent-driven interconnection protocol specifically at the Agent Gateway that can manage registration and routing based on semantic intent and dynamically advertised capabilities, rather than relying solely on fixed addresses or static identifiers. This protocol addresses these challenges by transforming the gateway from a static packet forwarder into a semantic ingress/egress point, enabling establishing agent interconnection while maintaining compatibility with the Interconnection Service Domain.

4.2. Representative Use Case

The following example illustrates a typical use case of the Intent-based Agent Interconnection Protocol operating at the Agent Gateway.

A Customer Service Dispatcher Agent, acting as the Leader Agent, transmits a user inquiry to the AG for intent resolution and service response. Specialized Partner Agents (Billing, Technical Support, Sales) have previously registered their specific semantic domains with the AG. For explicit queries like "My credit card was charged twice," the AG identifies a high-confidence semantic match and routes the request to the Billing Agent. Upon completion, the AG relays the result back to the Leader Agent. Conversely, when handling ambiguous inputs (e.g., "I'm having a bad experience") where semantic

similarity scores fail to meet the required confidence threshold, the AG encapsulates the intent and forwards the intent to a Generalist Fallback Agent. This mechanism ensures the request is constructively resolved via clarifying questions or human escalation, thereby preventing service failure.

5. Function Components

Figure 1 illustrates the interaction diagram between AI agent and Agent Gateway.

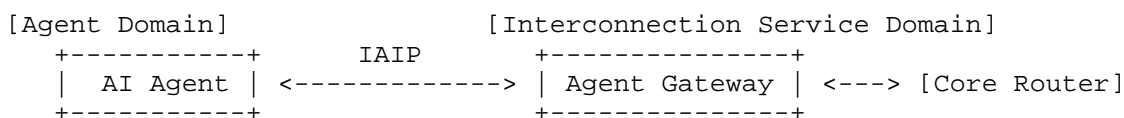


Figure 1: Interaction Diagram between AI Agent and Agent Gateway

To support the interaction with AI Agents, the AG MUST implement the following specific Functional Components within the Interconnection Service Domain.

- * ***Local Capability Registry (LCR):***The LCR maintains a dynamic database of attached Agents. The function of LCR is to store the Capability Profile of each registered agent. When an Agent sends the capability updating request, the LCR updates the mapping.
- * ***Agent Access Component (AAC):***The AAC serves as the termination point for the interconnection session between the AI Agent and the Agent Gateway. The function of AAC is to manage authentication, maintain session lifecycle, and verify message integrity. When an AI Agent initiates a connection or transmits a message, the AAC validates the agent's identity credentials and establishes the interconnection for message forwarding.
- * ***Intent Forwarding Routing (IFR):***The IFR serves as the dynamic routing decision engine within the Agent Gateway for forwarding intent-based requests. Its primary function is to map normalized intent vectors-derived from incoming task semantics-to a set of eligible Partner agents. Based on real-time capability matching scores, performance metrics (e.g., latency, success rate, and resource availability), and policy constraints, the IFR selects the optimal next-hop Partner agent for request forwarding.

6. Protocol Overview

The protocol functions in two primary phases.

- * *Phase 1: Agent Registration & Capability Advertisement.*This phase establishes the trust domain and service mesh. Both Leader Agents and Partner Agents MUST complete Identity Registration with the Agent Gateway (AG) to obtain the verifiable Agent Identity Code (AIC). Following identity registration, Partner Agents (or agents acting in a service-provisioning role) MUST additionally perform Capability Advertisement to publish their functional profiles to the Local Capability Registry (LCR).
- * *Phase 2: Intent-based Routing and Forwarding.*This phase establishes intent-based interconnection. The Leader Agent submits its intent. The AG resolves it against the LCR and IFR, selects suitable Partner Agents, and facilitates the connection.

Figure 2 illustrates the updated two-stage processing pipeline between AI Agents and AG.

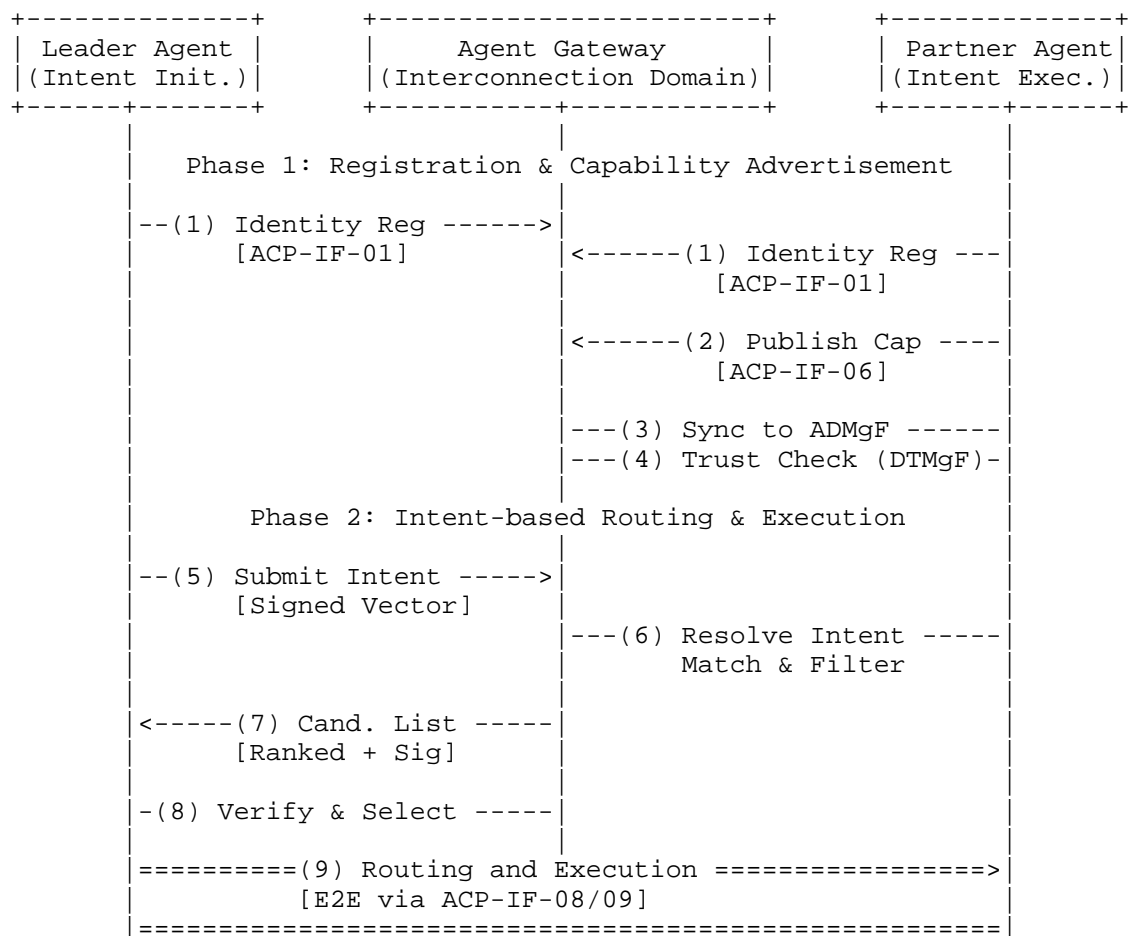


Figure 2: Two-Stage Processing Flow of IAIP

Phase 1: Agent Registration & Capability Advertisement

This stage is the prerequisite for any agent participation. It involves Identity Registration (mandatory for all) and Capability Publication (mandatory for service providers). The steps are as follows:

- * *Step1:*Identity Registration (Leader & Partner). Any Agent (whether Leader or Partner) MUST register its identity with the Agent Gateway using the ACP-IF-01 interface (between AIMF in the agent and AIMgF in the AG). This establishes a verifiable Agent Identity Code (AIC) and exchanges cryptographic credentials required for future intent signing (Leader) or service execution (Partner).
- * *Step2:*Capability Publication (Partner Agent). The Partner Agent publishes a structured capability description (e.g., supported task types, performance bounds, resource constraints) to the Agent Gateway via the ACP-IF-06 interface (ADMF <-> ADMgF). Note: If a Leader Agent also wishes to serve tasks, it MUST also perform this step.
- * *Step3:*Synchronization to Interconnection Service Domain. The Agent Gateway forwards the capability description to the Agent Description Management Function (ADMgF) in the Interconnection Service Domain. ADMgF persists the data and synchronizes it to the Agent Discovery Function (ADF) for indexing.
- * *Step4:*Trust Scope Enforcement. The Domain Trust Management Function (DTMgF) applies domain federation policies to determine whether the registered Agent is authorized to operate within the specific trust domain. Only authorized capabilities are made queryable via ADF.

Phase 2: Intent-based Routing and Forwarding

This stage is triggered when a registered Leader Agent submits a semantic intent request. The Agent Gateway resolves and routes the intent to eligible Partner Agents. The steps are as follows:

- * *Step5:*Intent Submission. The Leader Agent sends a signed semantic intent vector to the Agent Gateway. The signature MUST be generated using the credentials established in Step 1. The payload contains abstract task semantics.
- * *Step6:*Intent Resolution and Candidate Matching. The Agent Gateway invokes the ACP-IF-07 interface to query the ADF. The IFR (Intent Forwarding Routing Table) performs semantic matching against the LCR, filtering based on trust scopes, and ranking candidates.
- * *Step7:*Candidate Agents Return. The Gateway returns a ranked list of eligible Partner Agents to the Leader Agent.

- * ***Step8:*Leader-side Verification and Selection.** The Leader Agent verifies the gateway's signature and selects one or more Partners based on local policy.
- * ***Step9:*Routing and Execution.** The Leader Agent establishes a direct end-to-end session with the selected Partner Agent(s) using ACP-IF-08 for point-to-point interaction, or ACP-IF-09 for grouping-mode collaboration. All task payloads are transmitted over this encrypted channel.

7. Message Formats

This section specifies the binary structure of IAIP messages. The protocol uses a Type-Length-Value (TLV) design to ensure extensibility.

IAIP Message (TLV-based)		
Type	Len	Value
Common Header TLVs		
VERSION	1	Protocol version
MSG_TYPE	1	INTENT_REQ / CAP_ADV / ERROR
MSG_ID	var	Unique message identifier
SENDER_ID	var	Source agent / router ID
RECEIVER_ID	var	Target agent / router ID
TIMESTAMP	8	Unix time / logical time
TTL / EXPIRY	4	Validity / hop limit
CORRELATION_ID	var	Request-response correlation
INTEGRITY_TAG	var	MAC / signature
Payload TLVs (depend on MSG_TYPE)		
-- CAPABILITY ADVERTISEMENT (CAP_ADV) -----		
AGENT_ID	var	Advertising agent identifier
ENDPOINT	var	Network / logical endpoint
FUNC_DESC	var	Functional descriptors
INTENT_DOMAINS	var	Supported intent types
IO_SCHEMA	var	Input/output constraints
PERF_METRICS	var	Latency, success rate, etc.
RESOURCE_LIMITS	var	Token / compute budget

-- CAPABILITY REFRESH (CAP_REFRESH) -----			
AGENT_ID	var	Agent identifier	
TTL_UPDATE	4	Extended lifetime	
METRIC_DELTA	var	Updated performance metrics	
-- CAPABILITY DEREGISTER (CAP_DEREGISTER) -----			
AGENT_ID	var	Agent identifier	
REASON_CODE	2	Deregistration reason	
EFFECTIVE_TIME	8	Time of effect	
-- INTENT REQUEST (INTENT_REQ) -----			
OBJECTIVE	var	Task objective	
CONSTRAINTS	var	Preferences / limits	
CONTEXT	var	Context metadata	
PRIORITY	1	Request priority	
BUDGET	var	Cost / token budget	
PRIVACY_FLAG	1	Intent privacy level	
OBFUSCATED_INTENT	var	Encrypted / masked intent	
-- ROUTE DECISION / INTENT RESPONSE -----			
TARGET_AGENT_LIST	var	Selected target agent(s)	
MATCH_CONFIDENCE	var	Similarity / rank score	
FORWARDING_INFO	var	Next-hop / endpoint	
FALLBACK_INDIC.	1	Fallback used?	
-- ROUTING FEEDBACK -----			
OUTCOME	1	Success / failure	
OBSERVED_LATENCY	var	Execution latency	
OBSERVED_COST	var	Resource usage	
ERROR_INFO	var	Error details	
-- ERROR MESSAGE -----			
ERROR_CODE	2	Error identifier	
DIAGNOSTIC	var	Human-readable info	
RETRY_AFTER	4	Backoff hint	
-- SECURITY EXTENSION (SECURITY_EXT) -----			
AUTHN_INFO	var	Certificate / token ref	
AUTHZ_SCOPE	var	Authorization scope	
NONCE	var	Anti-replay	
SIGNATURE / MAC	var	Integrity protection	
ENCRYPTION_CTX	var	Intent confidentiality ctx	

Figure 3: IAIP message format

The table above summarizes the field requirements. Key fields include:

- *MSG_TYPE* Identifies the payload type (e.g., 0x01: CAP_ADV, 0x02: INTENT_REQ).
- *TTL / EXPIRY* Time-To-Live logic depending on message type. For routing messages (INTENT_REQ), this represents the hop limit. For registration messages, this represents the validity duration in seconds.
- *OBJECTIVE (in INTENT_REQ)* The semantic vector or structured description of the task.
- *RESOURCE_LIMITS (in CAP_ADV)* Specifies operational constraints, including token budget per request and computational cost.
- *FALLBACK_INDIC. (in RESPONSE)* Flag indicating if a generalist fallback agent was used (0x01) or a direct match was found (0x00).

8. Protocol Operation and Core Procedures

This section specifies the normative processing behavior of the Agent Gateway (AG), organized by the functional components defined in Section 5. The Agent Access Component (AAC) handles ingress connectivity and security; the Local Capability Registry (LCR) manages state; and the Intent Forwarding Routing Table (IFR) executes decision logic.

8.1. Illustrative Protocol Examples

The examples in this section are provided for illustration only and are non-normative.

8.1.1. Example A: Agent Registration (CAP_ADV)

A Partner Agent registers its translation service with cost constraints.

```
[Common Header] MSG_TYPE =  
    CAP_ADV SENDER_ID = "agent-trans-01", TTL/EXPIRY = 3600 [Payload],  
    FUNC_DESC = "service:translation; lang:en-to-zh", RESOURCE_LIMITS =  
    "max_tokens=2048", SECURITY_EXT = "(signature bytes...)"
```

8.1.2. Example B: Intent Resolution Flow

Step 1: Leader Agent sends Intent Request

```
[Common Header] MSG_TYPE =  
    INTENT_REQ SENDER_ID = "leader-agent-007", MSG_ID = "req-1024"  
    [Payload], OBJECTIVE = "(vector: [0.12, 0.95, ...])", BUDGET =  
    "100"
```

Step 2: Gateway returns Candidate List (Response)
The AG finds two matching agents. The first one is a direct match,
so Fallback Indicator is 0.

```
[Common Header] MSG_TYPE =  
    RESPONSE RECEIVER_ID = "leader-agent-007", CORRELATION_ID =  
    "req-1024" [Payload], TARGET_AGENT_LIST = [1] "agent-trans-01"  
    (Conf: 0.92, Endpoint: "10.0.1.5:8080"), [2] "agent-trans-03" (Conf:  
    0.85, Endpoint: "10.0.1.8:8080"), FALLBACK_INDIC. = 0x00 (No  
    Fallback)
```

8.1.3. Example C: Routing Feedback

After execution, the Leader Agent reports performance metrics back to
the AG to update the LCR's dynamic scores.

```
[Common Header] MSG_TYPE =  
    ROUTING_FEEDBACK SENDER_ID = "leader-agent-007" [Payload], OUTCOME =  
    1 (Success), OBSERVED_LATENCY = 250 (ms), OBSERVED_COST = 50  
    (tokens)
```

8.2. Agent Access Component (AAC) Operations

The AAC serves as the termination point for the interconnection
session. It MUST intercept all incoming IAIP messages before they
reach the LCR or IFR.

8.2.1. Session Authentication and Integrity

Upon receiving any message, the AAC MUST perform the following
validations:

- * *Identity Verification:* Validate the SENDER_ID against the
credentials provided in the AUTHN_INFO TLV or the underlying
transport session (e.g., mTLS).
- * *Message Integrity:* Verify the INTEGRITY_TAG or SIGNATURE. If
verification fails, the AAC MUST drop the message and return an
ERROR with code AUTH_FAILED.

8.2.2. Ingress Validation

The AAC acts as the first line of defense. It MUST validate the syntax of Common Header fields (e.g., VERSION, MSG_TYPE). Malformed packets MUST be rejected immediately to protect the internal IFR engine.

8.3. Local Capability Registry (LCR) Management

The LCR maintains the dynamic database of attached Agents. It processes capability management messages validated by the AAC.

8.3.1. Profile Creation and Updates

When the AAC forwards a CAP_ADV message, the LCR MUST:

1. Extract the AGENT_ID and map it to the provided ENDPOINT.
2. Store the FUNC_DESC, INTENT_DOMAINS, and RESOURCE_LIMITS as the agent's Capability Profile.
3. Set the entry's expiration time to Current_Time + TTL.

8.3.2. Lifecycle Maintenance

Upon receiving a CAP_REFRESH, the LCR MUST look up the existing profile and extend its validity by TTL_UPDATE.

The LCR MUST periodically purge entries that have exceeded their expiry time without refresh (STALE state) to ensure the IFR does not route to dead agents.

8.4. Intent Forwarding Routing Table (IFR) Decision Logic

The Intent Forwarding Routing Table (IFR) functions as the dynamic routing engine of the Agent Gateway. After an INTENT_REQ successfully passes AAC validation, the IFR MUST evaluate the request and produce a ranked TARGET_AGENT_LIST containing one or more eligible Partner Agents.

8.4.1. Candidate Retrieval

The IFR MUST interpret the OBJECTIVE TLV as an intent descriptor and query the Local Capability Registry (LCR) to obtain the current set of active Capability Profiles.

8.4.2. Constraint-Based Filtering

The IFR MUST apply mandatory constraints to reduce the candidate set prior to semantic evaluation. When a request includes constraint fields such as BUDGET, the IFR MUST compare these constraints against the corresponding attributes in the Capability Profiles (e.g., RESOURCE_LIMITS) and exclude agents that do not satisfy the required conditions.

Additional policy-based constraints MAY be applied according to local administrative policy.

8.4.3. Semantic Evaluation and Ranking

For the remaining candidates, the IFR MUST perform semantic matching between the request intent and the advertised capabilities. The specific matching method is implementation-dependent; however, the IFR MUST compute a relative confidence score that reflects the degree of intent-capability alignment.

The IFR MAY incorporate additional operational signals, such as historical performance metrics (e.g., latency or success rate), when deriving the final ranking.

The IFR MUST preserve the relative ordering of candidates when constructing the TARGET_AGENT_LIST.

8.4.4. Selection and Fallback

The IFR SHOULD select the highest-ranked candidates for inclusion in the response, subject to local policy.

If no candidate satisfies the acceptance criteria (e.g., minimum confidence threshold), the IFR SHOULD invoke the Fallback Mechanism. When fallback is applied, the Agent Gateway MUST set FALLBACK_INDIC to 0x01.

8.5. Forwarding and Loop Prevention

For multi-hop scenarios, the AG enforces loop prevention using the TTL field in the Common Header. If $TTL \leq 1$, the request is dropped. Duplicate MSG_ID, are suppressed to prevent broadcast storms.

8.6. Reliability and Error Handling

Errors detected by any component (AAC auth failure, LCR parsing error, IFR no-route) MUST result in an ERROR message returned to the sender, containing the appropriate ERROR_CODE.

8.7. Protocol Processing procedure at Agent Gateways

Figure 3 illustrates the protocol processing pipeline within the Agent Gateway (AG). The protocol operation relies on the coordination between three normative functional components: the Agent Access Component (AAC), the Local Capability Registry (LCR), and the Intent Forwarding Routing Table (IFR).

The process begins when a Partner Agent transmits a capability advertisement (CAP_ADV) [Section 8.1.1].

***Ingress Validation:** The AAC [Section 8.2] intercepts the message to perform session authentication and integrity verification.

***Registry Update:** Upon successful validation, the AAC forwards the payload to the LCR [Section 8.3]. The LCR parses the functional descriptors and updates its internal database, mapping the agent's semantic profile to its network endpoint.

The routing process is triggered when a Leader Agent submits an intent request (INTENT_REQ) [Section 8.1.2].

***Ingress Validation:** The AAC validates the request format and the sender's authorization scope.

***Routing Decision:** Validated requests are passed to the IFR [Section 8.4], which executes the core decision logic:

- * ***Candidate Retrieval:** The IFR queries the LCR [Section 8.4.1] to retrieve active agents capable of satisfying the intent objective.
- * ***Filtering:** Mandatory constraints (e.g., budget, latency) are applied to prune ineligible candidates [Section 8.4.2].
- * ***Ranking:** The IFR performs semantic evaluation [Section 8.4.3] to assign confidence scores to the remaining candidates.
- * ***Selection:** The eligible partner agents are selected based on local policy and fallback mechanisms [Section 8.4.4].

***Egress Processing:** The partner agent list is passed to the Forwarding module [Section 8.5]. This module enforces Time-To-Live (TTL) checks and duplicate suppression to prevent routing loops before dispatching the forwarded request to the Partner Agent.

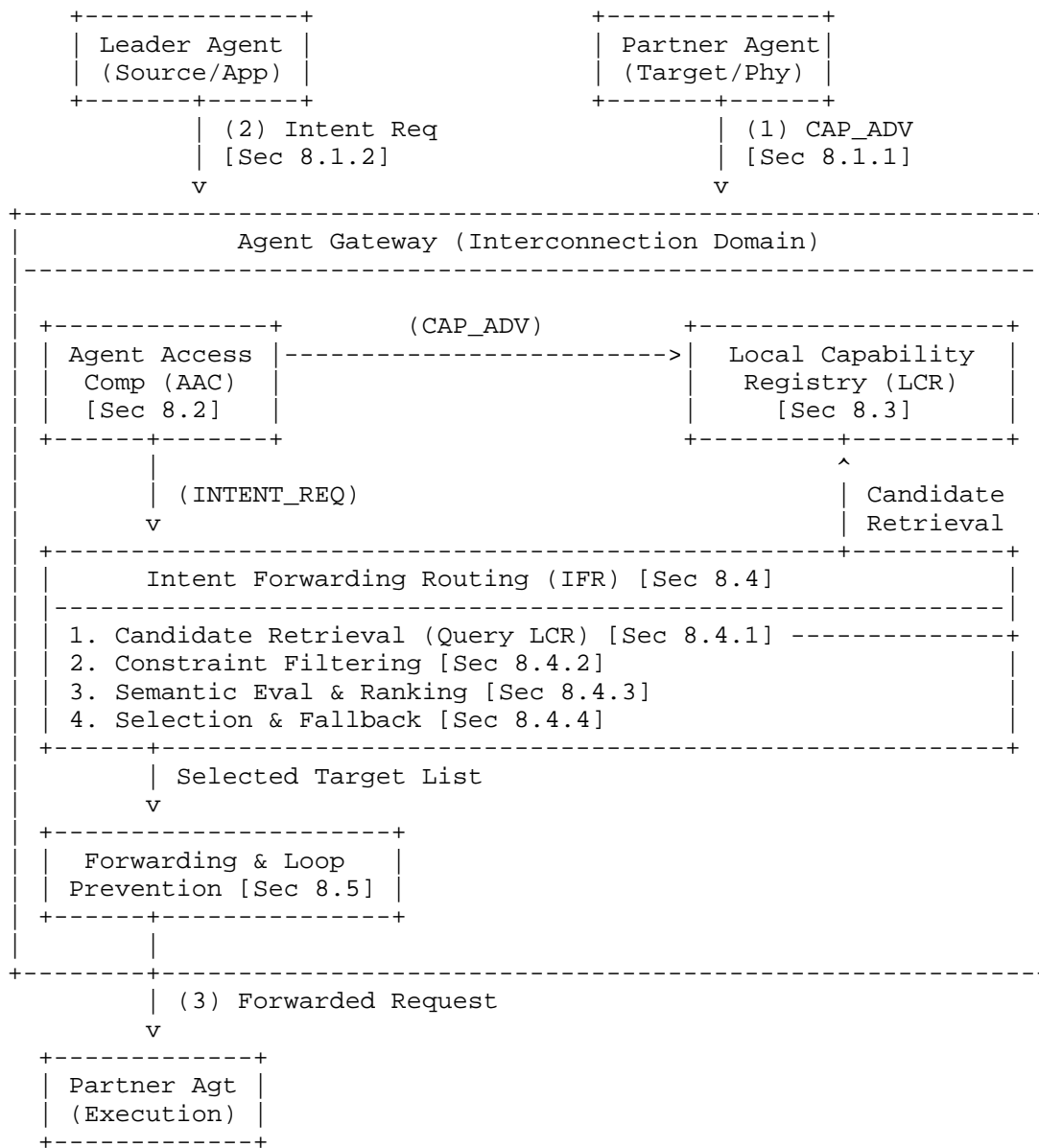


Figure 4: Protocol Processing procedure at Agent Gateways

9. Conclusions

This document specifies the functional components, protocol procedures, and message formats of the Intent-based Agent Interconnection Protocol (IAIP). It defines the mechanisms for capability-aware registration, semantic intent resolution, and dynamic routing. By abstracting the complexity of agent discovery and intent matching, IAIP ensures scalable and adaptive collaboration across heterogeneous agent systems.

10. Security Considerations

This document focuses on the Intent-based Agent Interconnection Protocol at the Agent Gateway for inter-agent collaboration in the Internet of Agents (IoA). Security of the IoA ecosystem is not detailed in this document. Security considerations relevant to cross-domain intent routing, capability advertisement integrity, trust federation among multiple agent service providers, and end-to-end confidentiality of agent interactions are suggested to be deeply discussed through other proposals.

11. IANA Considerations

TBD

12. Normative References

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

Authors' Addresses

Sheng Sun
ICT, CAS
Email: sunsheng@ict.ac.cn

Xinyi Zhang
CNIC, CAS
Email: xyzhang@cnic.cn