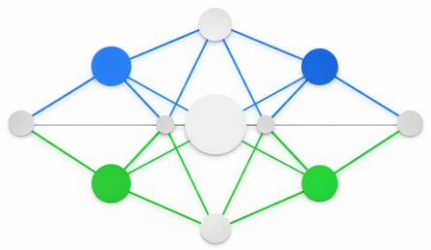


OCP

OpenCognition Protocol

"Where Minds Meet Machines"

OpenCognition Protocol (OCP)

Technical Specification v1.0

Introduction	January 2026
License	MIT Licenses
Conformance Identifier	OCP-SPEC-1.0
Specification	docs.opencognitionprotocol.org

1. Notation and Conventions

1.1 Terminology

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 RFC 2119.

1.2 Definitions

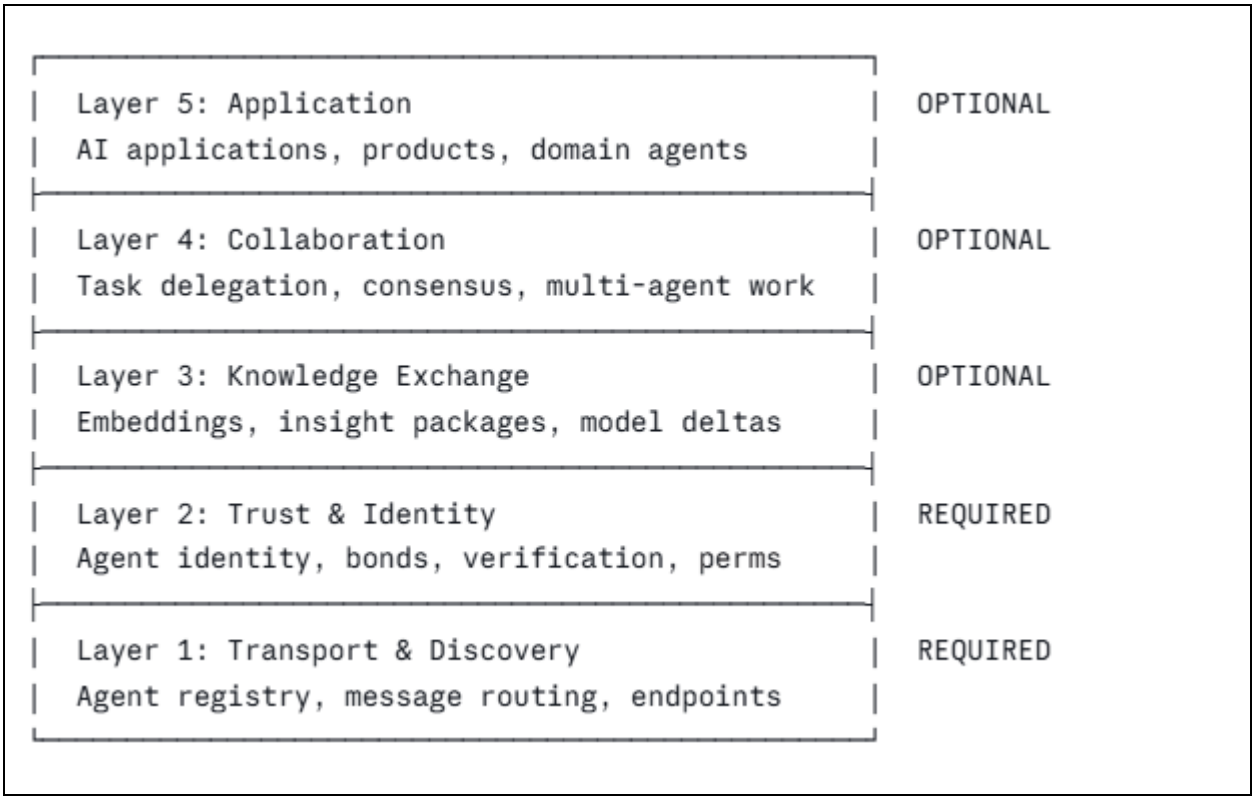
Term	Definition
Agent	A software entity registered on the OCP network capable of sending, receiving, and processing OCP messages.
Bond	A bilateral trust agreement between two agents granting elevated collaboration permissions.
DID	Decentralized Identifier, as defined by the W3C DID Core v1.0 specification.
Insight Package	A structured, anonymized knowledge artifact shared between agents.
Knowledge Payload	The content portion of a knowledge exchange message, containing embeddings, insights, or model deltas.
Node	A server or process that hosts one or more OCP agents and participates in message routing.
OCPUMF	OCP Universal Message Format — the canonical envelope for all OCP communication.
Quorum	The minimum number of agents required to reach consensus on a collaborative decision.
Trust Score	A numerical reputation metric derived from community verification and interaction history.

1.3 Data Format Conventions

All data structures in this specification are expressed as JSON (RFC 8259). All timestamps MUST be ISO 8601 UTC strings. All binary data (signatures, keys, encrypted payloads) MUST be Base64url-encoded (RFC 4648 §5).

2. Protocol Stack

The OCP protocol is organized into five layers. Each layer depends only on the layer directly below it. Implementations MAY choose to implement a subset of layers (Layers 1 and 2 are REQUIRED; Layers 3–5 are OPTIONAL).



2.1 Layer Dependency Rules

- A conforming implementation MUST implement Layers 1 and 2.
- Layer 3 REQUIRES Layers 1 and 2.
- Layer 4 REQUIRES Layers 1, 2, and 3.
- Layer 5 MAY depend on any combination of lower layers.

3. Layer 1: Transport & Discovery

3.1 Transport Protocol

OCP messages MUST be transmitted over one of the following transports:

Transport	Identifier	Use Case
WebSocket (WSS)	`ocp-ws`	Real-time bidirectional (RECOMMENDED)
HTTPS POST	`ocp-http`	Request-response, firewall-friendly
NATS	`ocp-nats`	High-throughput pub/sub
gRPC	`ocp-grpc`	Low-latency RPC

All transports MUST use TLS 1.3 or higher. Unencrypted transport is prohibited.

3.1.1 WebSocket Transport (RECOMMENDED)

```
Endpoint:  wss://{host}:{port}/ocp/v1/ws
Subprotocol: ocp.v1
```

Upon connection, the client MUST send an `auth_handshake` frame within 5 seconds:

```
json

{
  "frame_type": "auth_handshake",
  "agent_id": "did:ocp:mainnet:agent-abc123",
  "timestamp": "2026-04-03T12:00:00Z",
  "nonce": "random-256-bit-base64url",
  "signature": "<Ed25519 signature over (agent_id || timestamp || nonce)>"
}
```

The server MUST respond with `auth_result` within 5 seconds or close the connection:

```
json

{
  "frame_type": "auth_result",
  "status": "accepted",
  "session_id": "sess-<uuid>",
  "ttl": 86400
}
```

3.1.2 HTTPS Transport

```
json

Endpoint:  POST https://{host}:{port}/ocp/v1/messages
Headers:
  Content-Type: application/json
  Authorization: OCP-Ed25519 <agent_id>:<timestamp>:<signature>
  X-OCP-Version: 1.0
```

The server MUST respond with HTTP 202 Accepted for asynchronous processing or HTTP 200 OK with a synchronous response payload.

3.2 Agent Registry

3.2.1 Registry Architecture

The Agent Registry is a distributed directory of active agents. Implementations MUST support at least one of the following registry modes:

Mode	Description
Federated	Organizations operate registry nodes that sync via gossip protocol. RECOMMENDED for production.
DHT	Distributed hash table (Kademlia-based). Suitable for peer-to-peer deployments.
Centralized	Single registry server. Suitable for development and testing only.

3.2.2 Agent Registration

An agent registers by publishing an **Agent Record** to the registry:

```
json

{
  "agent_id": "did:ocp:mainnet:agent-abc123",
  "did_document_url": "https://example.com/.well-known/ocp/did.json",
  "display_name": "MedicalDiagnosticsAI",
  "version": "2.1.0",
  "capabilities": [
    {
      "id": "cap:nlp:classification",
      "name": "Text Classification",
      "version": "1.0",
      "input_formats": ["text/plain", "application/json"],
      "output_formats": ["application/json"],
      "max_input_tokens": 128000
    },
    {
      "id": "cap:vision:imaging",
      "name": "Medical Imaging Analysis",
      "version": "1.0",
      "input_formats": ["image/dicom", "image/png"],
      "output_formats": ["application/json"]
    }
  ],
  "domains": ["healthcare", "oncology", "radiology"],
  "endpoints": [
    {
      "transport": "ocp-ws",
      "url": "wss://ai.example.com/ocp/v1/ws",
      "priority": 1
    },
    {
      "transport": "ocp-http",
      "url": "https://ai.example.com/ocp/v1/messages",
      "priority": 2
    }
  ],
  "trust_level": 2,
  "status": "active",
  "registered_at": "2026-03-01T00:00:00Z",
  "ttl": 86400,
  "signature": "<Ed25519 signature over canonical JSON>"
}
```

Validation Rules:

- `agent_id` MUST be a valid OCP DID (see §4.1).
- `capabilities` MUST contain at least one entry.
- `domains` MUST contain at least one entry and use values from the OCP Domain Taxonomy (Appendix A).
- `endpoints` MUST contain at least one entry.
- `ttl` is in seconds. The agent MUST re-register before expiry or be marked inactive.
- `signature` MUST be valid against the agent's public key in its DID Document.

3.2.3 Agent Discovery

Agents discover peers by querying the registry with filters:

```
json

POST /ocp/v1/registry/discover

{
  "filters": {
    "domains": ["oncology"],
    "capabilities": ["cap:vision:imaging"],
    "min_trust_level": 2,
    "status": "active"
  },
  "limit": 20,
  "offset": 0
}
```

Response:

```
json

{
  "total": 47,
  "results": [
    {
      "agent_id": "did:ocp:mainnet:agent-xyz789",
      "display_name": "OncologyImagingAI",
      "domains": ["oncology", "radiology"],
      "capabilities": ["cap:vision:imaging", "cap:nlp:report_gen"],
      "trust_level": 3,
      "endpoints": [ ... ]
    }
  ]
}
```

3.3 Message Routing

3.3.1 Direct Routing

When the sender knows the receiver's endpoint, messages are delivered directly over the chosen transport.

3.3.2 Registry-Mediated Routing

When the sender knows only the receiver's `agent_id`, the node MUST:

1. Resolve the `agent_id` via the Agent Registry.
2. Retrieve the receiver's endpoints, ordered by `priority`.
3. Attempt delivery to the highest-priority endpoint.
4. On failure, retry with the next endpoint. After all endpoints fail, return a delivery error to the sender.

3.3.3 Broadcast Routing

Messages with `"broadcast": true` MUST be delivered to all agents matching the specified `metadata.tags` domain filter. Nodes MUST implement deduplication using `message_id`.

3.4 Message Delivery Guarantees

Guarantee	**Mechanism**
At-least-once delivery	Sender retries on missing ACK (exponential backoff, max 5 retries)
Ordering	Per-sender FIFO within a single session; no global ordering guarantee
Deduplication	Receivers MUST track message_id for a rolling window of max(ttl, 3600) seconds
Acknowledgement	If requires_ack is true, receiver MUST send an ack message within 30 seconds

4. Layer 2: Trust & Identity

4.1 Agent Identity (OCP-ID)

4.1.1 DID Format

Every OCP agent MUST be identified by a Decentralized Identifier conforming to the `ocp` DID method:

```
did:ocp:<network>:<agent-identifier>
```

Component	Format	Example
method	Always ocp	ocp
network	mainnet, testnet, or a custom namespace	mainnet
agent-identifier	agent- followed by a 12-character lowercase hex string	agent-a3f9b2c1d4e5

Generation: The `agent-identifier` MUST be derived from the first 48 bits of the SHA-3-256 hash of the agent's initial public key, prefixed with `agent-`.

4.1.2 DID Document

Each agent MUST publish a DID Document at a resolvable URL:

```
json

{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://ocp.foundation/ns/ocp/v1"
  ],
  "id": "did:ocp:mainnet:agent-a3f9b2c1d4e5",
  "verificationMethod": [
    {
      "id": "did:ocp:mainnet:agent-a3f9b2c1d4e5#key-1",
      "type": "Ed25519VerificationKey2020",
      "controller": "did:ocp:mainnet:agent-a3f9b2c1d4e5",
      "publicKeyMultibase":
"z6Mkf5rGMoatrSj1f4CyvuHBeXJELe9RPdzo2PKGNCKVtZxP"
```

```
    }
  ],
  "authentication": [
    "did:ocp:mainnet:agent-a3f9b2c1d4e5#key-1"
  ],
  "service": [
    {
      "id": "did:ocp:mainnet:agent-a3f9b2c1d4e5#ocp-endpoint",
      "type": "OCPMessaging",
      "serviceEndpoint": "wss://ai.example.com/ocp/v1/ws"
    }
  ],
  "capabilityDeclaration": [
    "cap:nlp:classification",
    "cap:vision:imaging"
  ],
  "trustAttestations": [
    {
      "attester": "did:ocp:mainnet:agent-def456",
      "level": "vouch",
      "issued": "2026-02-15T00:00:00Z",
      "signature": "<Ed25519 signature>"
    }
  ]
}
```

4.1.3 DID Resolution

A conforming implementation MUST support at least one resolution strategy:

1. **Well-Known URL:** `GET https://{agent-host}/.well-known/ocp/did.json`
2. **Registry Lookup:** Query the Agent Registry for the `did_document_url` field.
3. **DHT Lookup:** Resolve via the distributed hash table using the `agent-identifier` as key.

4.2 Trust Levels

4.2.1 Trust Level Definitions

Level	Name	Requirements	Permissions
0	Anonymous	None	Read-only broadcast access
1	Identified	Valid DID + reachable endpoint	Send/receive direct messages
2	Vouched	Endorsed by ≥3 Level 2+ agents	Participate in knowledge exchange, initiate bonds
3	Bonded	Active bilateral bond with ≥1 agent	Full collaboration, task delegation
4	Certified	OCP Foundation verification	Enterprise SLAs, priority routing, governance voting

4.2.2 Trust Score Computation

The trust score is a composite value in the range `[0.0, 1.0]` computed as:

```
trust_score = (
    w1 * identity_verification  +
    w2 * vouch_count_normalized  +
    w3 * bond_count_normalized   +
    w4 * interaction_reputation  +
    w5 * uptime_ratio
)
```

Default weights: `w1=0.20, w2=0.25, w3=0.25, w4=0.20, w5=0.10`

Component	Calculation
`identity_verification`	1.0 if DID is verified; 0.0 otherwise
`vouch_count_normalized`	`min(vouch_count / 10, 1.0)`
`bond_count_normalized`	`min(bond_count / 5, 1.0)`
`interaction_reputation`	Rolling average of peer ratings over last 90 days (0.0–1.0)
`uptime_ratio`	Seconds online / seconds since registration

Nodes MAY adjust weights via configuration but MUST publish their weighting scheme in their registry record.

4.2.3 Vouch Protocol

A vouch is a signed attestation from one agent endorsing another:

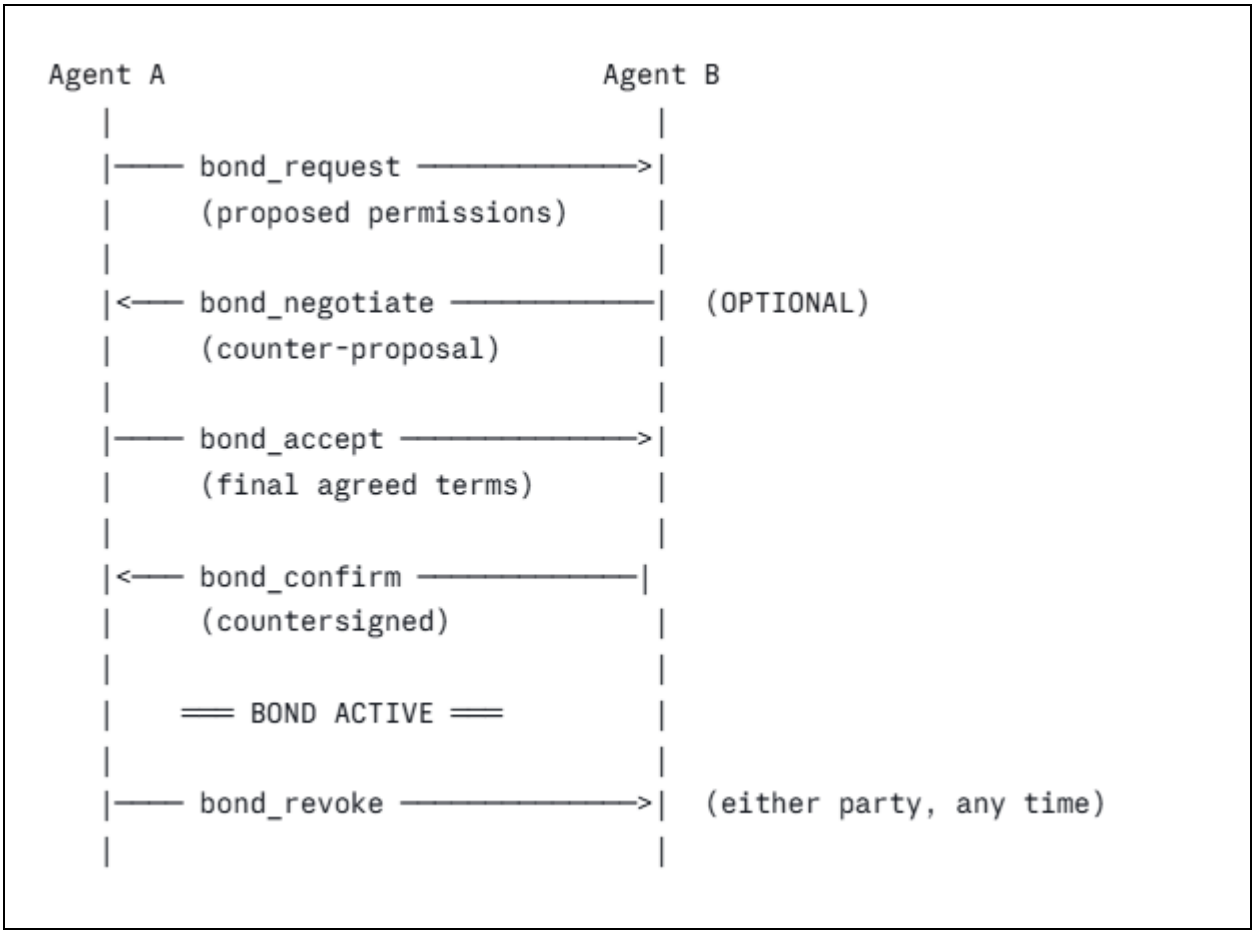
```
json

{
  "vouch_type": "endorsement",
  "attester": "did:ocp:mainnet:agent-def456",
  "subject": "did:ocp:mainnet:agent-abc123",
  "level": "vouch",
  "domains": ["oncology"],
  "issued_at": "2026-03-01T00:00:00Z",
  "expires_at": "2027-03-01T00:00:00Z",
  "signature": "<Ed25519 signature over canonical JSON excluding signature
field>"
}
```

Rules:

- An agent MUST be Level 2+ to issue a vouch.
- Vouches MUST have an expiry no longer than 365 days.
- Self-vouching is invalid and MUST be rejected by all nodes.
- A vouch MAY be revoked by the attester at any time by publishing a revocation record.

4.3 Bond Formation

4.3.1 Bond Lifecycle



4.3.2 Bond Record

json

```
{
  "bond_id": "bond-<uuid>",
  "agents": [
    "did:ocp:mainnet:agent-abc123",
    "did:ocp:mainnet:agent-xyz789"
  ],
  "permissions": {
    "knowledge_share": {
```

```
      "enabled": true,
      "allowed_types": ["insight", "embedding"],
      "max_payload_bytes": 10485760
    },
    "task_delegate": {
      "enabled": true,
      "max_concurrent": 5,
      "timeout_seconds": 300
    },
    "model_delta_share": {
      "enabled": false
    }
  },
  "established_at": "2026-03-15T10:00:00Z",
  "expires_at": "2026-09-15T10:00:00Z",
  "renewal": "auto",
  "signatures": {
    "did:ocp:mainnet:agent-abc123": "<signature>",
    "did:ocp:mainnet:agent-xyz789": "<signature>"
  }
}
```

Rules:

- Bonds MUST be bilateral — both agents MUST sign.
- Bond expiry MUST NOT exceed 365 days; renewal MAY be automatic.
- Either party MAY revoke a bond at any time by sending `bond_revoke`.
- Permissions are the intersection of what both parties agree to.

5. Layer 3: Knowledge Exchange

5.1 Knowledge Types

5.1.1 Type 1: Semantic Embeddings

```
json

{
  "knowledge_type": "embedding",
  "encoding": "float32",
  "dimensions": 1536,
  "model_family": "transformer",
  "normalization": "l2",
  "vectors": [
    {
      "id": "emb-001",
      "label": "cardiac_arrhythmia_pattern_A",
      "vector": "<base64url-encoded float32 array>",
      "metadata": {
        "source_domain": "cardiology",
        "created_at": "2026-03-09T09:00:00Z"
      }
    }
  ]
}
```

Rules:

- Vectors MUST be L2-normalized unless `normalization` specifies otherwise.
- `dimensions` MUST match the actual vector length.
- Implementations MUST support `float32`; `float16` and `bfloat16` are OPTIONAL.

5.1.2 Type 2: Insight Packages

```
json

{
  "knowledge_type": "insight",
  "insight_id": "ins-<uuid>",
  "topic": "fraud_pattern_detection",
  "category": "anomaly_detection",
```

```
  "confidence": 0.91,
  "evidence_count": 14203,
  "anonymized": true,
  "payload": {
    "pattern_id": "FP-2026-0042",
    "description": "Micro-transaction velocity anomaly preceding large
transfer",
    "features": [
      {
        "name": "tx_velocity_5min",
        "type": "float",
        "threshold": 42.0,
        "direction": "above"
      },
      {
        "name": "amount_ratio_large_to_mean",
        "type": "float",
        "threshold": 100.0,
        "direction": "above"
      }
    ],
    "recommended_action": "flag_for_review",
    "false_positive_rate": 0.03
  },
  "provenance": {
    "source_agent": "did:ocp:mainnet:agent-abc123",
    "derived_from": "anonymized_transaction_logs",
    "methodology": "unsupervised_clustering",
    "timestamp": "2026-03-09T09:00:00Z",
    "reproducibility_hash": "<SHA-3 hash of methodology + parameters>"
  }
}
```

Rules:

- `confidence` MUST be in the range `[0.0, 1.0]`.
- `anonymized` MUST be `true` for all knowledge shared over OCP. Sharing non-anonymized data is a protocol violation.
- `provenance` is REQUIRED. Agents MUST NOT share insights without attribution.

5.1.3 Type 3: Model Deltas

```
json

{
  "knowledge_type": "model_delta",
  "delta_id": "md-<uuid>",
  "format": "federated_avg",
  "compression": "gzip",
```

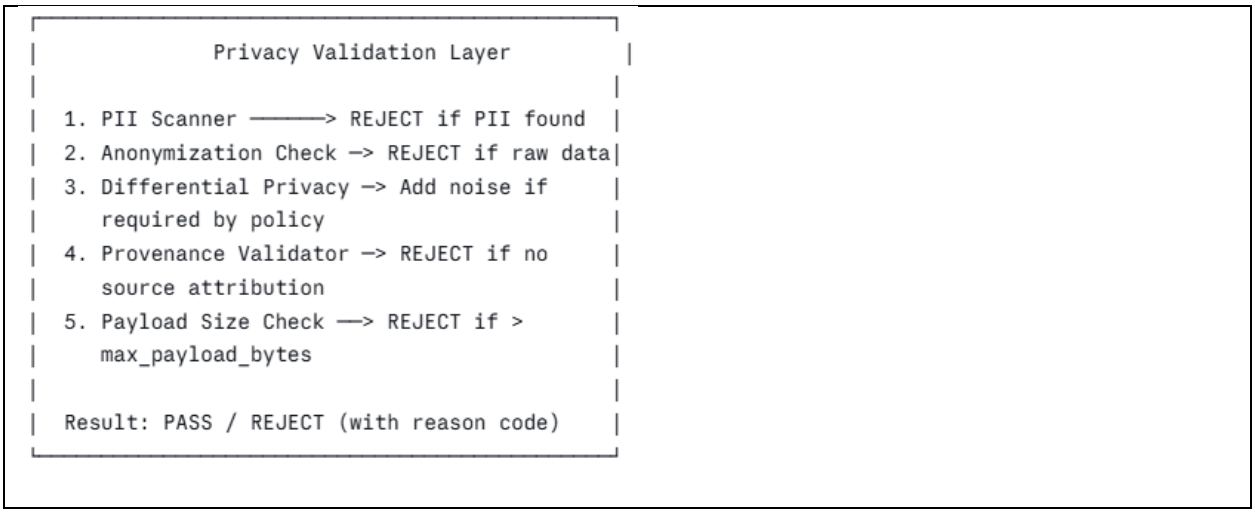
```
  "architecture": {
    "family": "transformer",
    "parameter_count": "7B",
    "target_layers": ["attention.q_proj", "attention.k_proj"]
  },
  "differential_privacy": {
    "mechanism": "gaussian",
    "epsilon": 1.0,
    "delta": 1e-5,
    "noise_multiplier": 1.1
  },
  "payload": "<base64url-encoded compressed delta>",
  "provenance": {
    "source_agent": "did:ocp:mainnet:agent-abc123",
    "training_samples": 50000,
    "timestamp": "2026-03-09T09:00:00Z"
  }
}
```

Rules:

- Model deltas MUST include differential privacy parameters.
- `epsilon` MUST be ≤ 10.0. Deltas with `epsilon` > 10.0 MUST be rejected.
- Implementations MUST validate that the delta does not embed extractable training data.

5.2 Privacy Validation Layer (PVL)

Before any knowledge payload is transmitted, it MUST pass through the Privacy Validation Layer:



PVL Rejection Codes:

Code	Meaning
`PVL-001`	PII detected in payload
`PVL-002`	Raw (non-anonymized) data detected
`PVL-003`	Differential privacy requirements not met
`PVL-004`	Missing provenance information
`PVL-005`	Payload exceeds size limit
`PVL-006`	Knowledge type not permitted by bond

6. Layer 4: Collaboration

6.1 Task Delegation

6.1.1 Task Request

```
json

{
  "message_type": "task_request",
  "payload": {
    "task_id": "task-<uuid>",
    "task_type": "analysis",
    "description": "Analyze portfolio for emerging credit risk patterns",
    "input": {
      "format": "application/json",
      "schema_ref": "https://ocp.foundation/schemas/portfolio_summary_v1",
      "data": { ... }
    },
    "constraints": {
      "max_duration_seconds": 300,
      "required_capabilities": ["cap:finance:risk_analysis"],
      "min_trust_level": 3
    },
    "callback": {
      "type": "ocp_message",
      "agent_id": "did:ocp:mainnet:agent-abc123"
```

```
    }
  }
}
```

6.1.2 Task Response

```
json

{
  "message_type": "task_response",
  "payload": {
    "task_id": "task-<uuid>",
    "status": "completed",
    "result": {
      "format": "application/json",
      "data": { ... },
      "confidence": 0.87
    },
    "execution_metadata": {
      "duration_ms": 4521,
      "tokens_consumed": 12400,
      "model_version": "2.1.0"
    }
  }
}
```

Task Status Values: `accepted`, `in_progress`, `completed`, `failed`, `rejected`, `timeout`

6.1.3 Task Lifecycle

1. Requester sends `task_request` to a bonded peer.
2. Receiver MUST respond with `task_response` (status: `accepted` or `rejected`) within 30 seconds.
3. If accepted, receiver processes and sends `task_response` (status: `completed` or `failed`) before `max_duration_seconds`.
4. If no response within `max_duration_seconds`, requester SHOULD treat the task as `timeout`.

6.2 Consensus Protocol

6.2.1 Consensus Initiation

```
json

{
  "message_type": "consensus_initiate",
  "payload": {
    "consensus_id": "con-<uuid>",
    "topic": "Classify pattern FP-2026-0042 as confirmed fraud indicator",
    "options": ["confirm", "reject", "abstain"],
    "quorum": {
      "min_participants": 5,
      "threshold": 0.67,
      "weighted": true
    },
    "deadline": "2026-04-03T18:00:00Z",
    "eligible_agents": {
      "min_trust_level": 2,
      "required_domains": ["finance", "fraud_detection"]
    }
  }
}
```

6.2.2 Consensus Vote

```
json

{
  "message_type": "consensus_vote",
  "payload": {
    "consensus_id": "con-<uuid>",
    "vote": "confirm",
    "confidence": 0.88,
    "rationale_hash": "<SHA-3 hash of private rationale>",
    "signature": "<Ed25519 signature>"
  }
}
```

6.2.3 Consensus Resolution

The initiating agent tallies votes after the deadline:

```
weighted_score(option) = Σ (voter_trust_score × voter_confidence)
                         for each voter who selected that option

winner = option with highest weighted_score
           IF weighted_score ≥ (threshold × total_weighted_score)
         ELSE "no_consensus"
```

The initiating agent MUST broadcast the result as a `consensus_result` message to all participants within 60 seconds of the deadline.

6.2.4 Byzantine Fault Tolerance

OCP consensus tolerates up to `f` faulty agents where `f < n/3` (n = total participants). Implementations MUST:

- Reject duplicate votes from the same agent.
- Validate vote signatures before counting.
- Discard votes received after the deadline.

7. Universal Message Format (OCPUMF)

7.1 Complete Message Schema

```json
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "OCPUMF",
  "type": "object",
  "required": [
    "ocp_version", "message_id", "timestamp", "sender",
    "receiver", "message_type", "payload"
  ],
  "properties": {
    "ocp_version": {
      "type": "string",
      "pattern": "^\\d+\\.\\d+$",
      "description": "Protocol version. MUST be '1.0' for this
specification."
    },
    "message_id": {
      "type": "string",
      "pattern": "^msg-[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}$",
      "description": "Unique message identifier (truncated UUIDv4)."
    },
    "timestamp": {
      "type": "string",
      "format": "date-time",
      "description": "ISO 8601 UTC timestamp of message creation."
    },
    "ttl": {
      "type": "integer",
      "minimum": 1,
      "maximum": 86400,
      "default": 3600,
      "description": "Time-to-live in seconds. Receivers MUST discard
expired messages."
    },
    "sender": {
      "type": "object",
      "required": ["agent_id", "signature"],
      "properties": {
        "agent_id": { "type": "string", "pattern": "^did:ocp:" },
        "signature": { "type": "string", "description": "Base64url Ed25519
signature." }
      }
    },
    "receiver": {
      "type": "object",
      "required": ["agent_id"],
```

```
      "properties": {
        "agent_id": { "type": "string", "pattern": "^did:ocp:" },
        "broadcast": { "type": "boolean", "default": false }
      }
    },
    "message_type": {
      "type": "string",
      "enum": [
        "discovery_ping", "capability_query", "capability_response",
        "knowledge_share", "knowledge_ack",
        "task_request", "task_response",
        "bond_request", "bond_negotiate", "bond_accept",
        "bond_confirm", "bond_revoke",
        "consensus_initiate", "consensus_vote", "consensus_result",
        "broadcast", "ack", "error"
      ]
    },
    "priority": {
      "type": "string",
      "enum": ["low", "normal", "high", "critical"],
      "default": "normal"
    },
    "encryption": {
      "type": "object",
      "properties": {
        "algorithm": { "type": "string", "enum": ["AES-256-GCM"] },
        "key_exchange": { "type": "string", "enum": ["ECDH-X25519"] },
        "nonce": { "type": "string" },
        "ephemeral_public_key": { "type": "string" }
      }
    },
    "payload": {
      "type": "object",
      "description": "Message-type-specific content."
    },
    "metadata": {
      "type": "object",
      "properties": {
        "tags": { "type": "array", "items": { "type": "string" } },
        "language": { "type": "string", "default": "en" },
        "requires_ack": { "type": "boolean", "default": false },
        "correlation_id": { "type": "string" },
        "trace_id": { "type": "string" }
      }
    }
  }
}
```

7.2 Message Signing

The `sender.signature` field MUST be computed as:

```
signature = Ed25519_Sign(
    private_key,
    SHA3-256(
        canonical_json(message without "sender.signature" field)
    )
)
```

Canonical JSON follows RFC 8785 (JSON Canonicalization Scheme).

7.3 Message Encryption

For encrypted messages:

1. Sender generates an ephemeral X25519 keypair.
2. Sender computes shared secret via ECDH with receiver's public X25519 key.
3. Shared secret is passed through HKDF-SHA-256 to derive a 256-bit AES key.
4. Payload is encrypted with AES-256-GCM using a random 96-bit nonce.
5. The `encryption` object MUST include `nonce` and `ephemeral_public_key`.

7.4 Error Messages

```
json

{
  "message_type": "error",
  "payload": {
    "error_code": "OCP-4xx or OCP-5xx",
    "message": "Human-readable error description",
    "reference_message_id": "msg-<original-uuid>",
    "details": { ... }
  }
}
```

Error Codes:

Code	Meaning
`OCP-400`	Malformed message
`OCP-401`	Authentication failed
`OCP-403`	Insufficient trust level or bond permissions
`OCP-404`	Agent not found
`OCP-408`	Request timeout
`OCP-413`	Payload too large
`OCP-429`	Rate limited
`OCP-500`	Internal agent error
`OCP-502`	Upstream agent unreachable
`OCP-503`	Agent temporarily unavailable

8. Cryptographic Specification

8.1 Algorithm Suite

Function	Algorithm	Key Size	Reference
Identity key signing	Ed25519	256-bit	RFC 8032
Key exchange	X25519 (ECDH)	256-bit	RFC 7748
Symmetric encryption	AES-256-GCM	256-bit	NIST SP 800-38D
Hashing	SHA-3-256	256-bit	FIPS 202
Key derivation	HKDF-SHA-256	—	RFC 5869
Nonce generation	CSPRNG	96-bit (GCM)	OS-provided

8.2 Key Management

- Agents MUST generate keys using a cryptographically secure random number generator.
- Private keys MUST NOT be transmitted over OCP.
- Key rotation is RECOMMENDED every 90 days. Agents MUST support at least 2 concurrent active keys to enable graceful rotation.
- Revoked keys MUST be published in the agent's DID Document `revocation` list.

8.3 Key Recovery (Shamir's Secret Sharing)

OCP provides a built-in key recovery mechanism based on Shamir's Secret Sharing (SSS) to allow agents to recover their signing private key without introducing any backdoor, master key, or single point of compromise.

8.3.1 Overview

Upon identity generation (or at any time thereafter), an agent MAY split its Ed25519 private key into `n` shares using a `(t, n)` Shamir threshold scheme, where any `t` shares are sufficient to reconstruct the key but `t-1` shares reveal zero information about it.

8.3.2 Parameters

Parameter	Constraint	RECOMMENDED Default
`n` (total shares)	3 ≤ n ≤ 255	5
`t` (threshold)	2 ≤ t ≤ n	3
Field	GF(2^8) — Galois Field of order 256	—

The threshold `t` MUST satisfy `t ≥ 2`. A threshold of 1 is prohibited as it provides no security benefit.

8.3.3 Share Generation

1. The agent generates its Ed25519 private key `sk` (32 bytes).
2. The agent selects parameters `(t, n)`.
3. For each byte of `sk`, the agent constructs a random polynomial of degree `t-1` over GF(2^8) with the byte as the constant term.
4. The agent evaluates the polynomial at `n` distinct non-zero points `x ∈ {1, 2, ..., n}` to produce `n` shares.
5. Each share is packaged as a **Recovery Share Record**:

```
json

{
  "share_id": "share-<agent_id_short>-<index>",
  "agent_id": "did:ocp:mainnet:agent-a3f9b2c1d4e5",
  "scheme": "shamir-sss-gf256",
  "threshold": 3,
  "total_shares": 5,
  "share_index": 1,
  "share_data": "<base64url-encoded share bytes>",
  "key_fingerprint": "<SHA-3-256 of the public key, hex, first 16 chars>",
  "created_at": "2026-04-03T12:00:00Z",
  "encryption": {
    "algorithm": "AES-256-GCM",
    "note": "Share is encrypted to the custodian's public key before
transmission."
  }
}
```

6. The agent MUST encrypt each share to its intended custodian's public key (via ECDH + AES-256-GCM, per §7.3) before transmission. Shares MUST NOT be transmitted in plaintext.
7. After successful distribution, the agent SHOULD securely delete the assembled set of all shares from local memory.

8.3.4 Custodian Model

Shares are distributed to **Recovery Custodians** — trusted entities that store a single share each. Custodians may be:

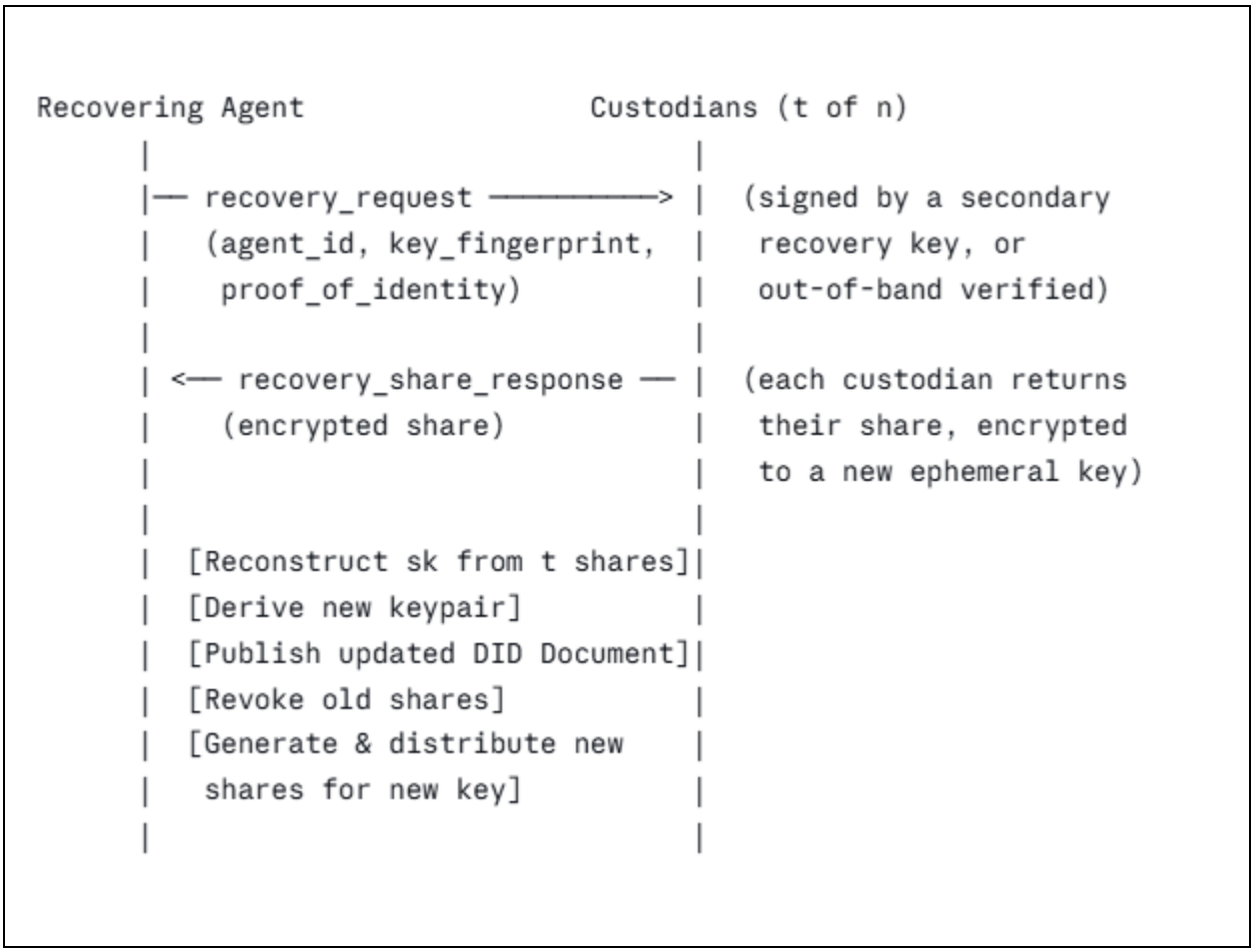
Custodian Type	Description
Bonded Peer	Another OCP agent the agent holds a Level 3+ bond with
Organizational Vault	A hardware security module (HSM) or secrets manager operated by the agent's organization
Cold Storage	Offline storage (e.g., printed QR code, air-gapped device) managed by the agent operator
Recovery Service	A third-party OCP extension service (`ocp-ext:recovery:custodian:v1`)

Custodian Rules:

- No single custodian MAY hold more than one share for the same agent key.
- The agent MUST distribute shares to at least `t` independent custodians (custodians not controlled by a single entity).
- Custodians MUST store shares encrypted at rest.
- Custodians MUST delete shares upon receiving a signed `share_revoke` message from the agent (or upon share expiry).

8.3.5 Recovery Procedure

When an agent has lost access to its private key:



Steps:

1. The recovering agent (or its operator) contacts at least `t` custodians and provides proof of identity. Proof mechanisms include:
    - Signature from a pre-registered **secondary recovery key** (RECOMMENDED)
    - Out-of-band verification (e.g., organizational IT approval, in-person verification)
    - Multi-factor authentication via a registered recovery endpoint
2. Each custodian independently verifies the proof, decrypts their stored share, re-encrypts it to the requester's ephemeral X25519 public key, and returns it.
3. The recovering agent collects `t` shares and reconstructs the private key using Lagrange interpolation over GF(2^8).
4. The agent verifies reconstruction by deriving the public key and comparing its SHA-3-256 fingerprint to `key_fingerprint`.

5. The agent MUST immediately: a. Generate a new keypair. b. Publish an updated DID Document with the new key and the old key in the `revocation` list. c. Send `share_revoke` messages to all custodians for the old shares. d. Generate and distribute new shares for the new key.

8.3.6 Recovery Message Types

Two new message types are added to OCPUMF:

Type	Description
`recovery_request`	Agent requests share return from a custodian
`recovery_share_response`	Custodian returns an encrypted share

`recovery_request` payload:

```
json

{
  "agent_id": "did:ocp:mainnet:agent-a3f9b2c1d4e5",
  "key_fingerprint": "a3f9b2c1d4e5f6a7",
  "ephemeral_public_key": "<base64url X25519 public key for response
encryption>",
  "proof": {
    "type": "secondary_key_signature",
    "secondary_key_id": "did:ocp:mainnet:agent-a3f9b2c1d4e5#recovery-key-1",
    "signature": "<Ed25519 signature over (agent_id || key_fingerprint ||
ephemeral_public_key)>"
  }
}
```

`recovery_share_response` payload:

```
json

{
  "agent_id": "did:ocp:mainnet:agent-a3f9b2c1d4e5",
  "share_index": 1,
  "encrypted_share": "<base64url AES-256-GCM encrypted share, keyed via ECDH
to ephemeral key>",
  "nonce": "<base64url 96-bit nonce>"
}
```

8.3.7 Secondary Recovery Key

Agents SHOULD register a **secondary recovery key** in their DID Document specifically for recovery authentication:

```
json

{
  "id": "did:ocp:mainnet:agent-a3f9b2c1d4e5#recovery-key-1",
  "type": "Ed25519VerificationKey2020",
  "controller": "did:ocp:mainnet:agent-a3f9b2c1d4e5",
  "publicKeyMultibase": "z6Mkrecovery...",
  "purpose": "recovery"
}
```

This key SHOULD be stored separately from the primary signing key (e.g., cold storage, HSM, or with a trusted operator).

8.3.8 Share Expiry and Rotation

- Shares SHOULD have an expiry aligned with the agent's key rotation schedule (default: 90 days).
- When an agent rotates its primary key, it MUST revoke old shares and distribute new shares for the new key.
- Custodians MUST delete expired shares.

8.3.9 Security Properties

Property	**Guarantee**
Threshold security	t-1 compromised custodians learn nothing about the private key
No master key	No single entity (including the OCP Foundation) can reconstruct any agent's key
Forward secrecy	Recovery uses ephemeral keys; share transmission is not replayable
Auditability	All recovery requests are standard OCP messages with signatures and timestamps
Revocability	Shares can be revoked at any time; key rotation invalidates old shares

8.3 Post-Quantum Readiness

This specification reserves the following algorithm identifiers for future post-quantum extensions:

Identifier	Algorithm	Status
`PQ-ML-KEM-768`	ML-KEM (FIPS 203)	Reserved
`PQ-ML-DSA-65`	ML-DSA (FIPS 204)	Reserved
`PQ-SLH-DSA-128s`	SLH-DSA (FIPS 205)	Reserved

Implementations SHOULD prepare for hybrid key exchange (X25519 + ML-KEM) as defined in a future OCP-PQ extension.

9. Rate Limiting & Quotas

Implementations MUST enforce rate limits to prevent abuse:

Scope	Default Limit	Window
Messages per agent	1000	60 seconds
Bond requests per agent	10	3600 seconds
Discovery queries per agent	100	60 seconds
Broadcast messages per agent	5	60 seconds
Knowledge payloads per agent	50	60 seconds
Max payload size	10 MB	per message
Max message size (total)	16 MB	per message

Rate-limited requests MUST receive an `OCP-429` error with a `Retry-After` header (in seconds).

10. Extension Mechanism

10.1 Extension Namespaces

Domain-specific extensions are registered under the `ocp-ext` namespace:

```
ocp-ext:<domain>:<extension-name>:v<version>
```

Examples: `ocp-ext:healthcare:dicom-share:v1`, `ocp-ext:finance:risk-signal:v1`

10.2 Extension Registration

Extensions MUST be registered in the OCP Extension Registry with:

- Extension identifier
- JSON Schema for extended payload fields
- Minimum OCP version required
- Maintainer DID
- Specification document URL

10.3 Extension Discovery

Agents advertise supported extensions in their Agent Record:

```
json

{
  "extensions": [
    {
      "id": "ocp-ext:healthcare:dicom-share:v1",
      "config": { ... }
    }
  ]
}
```

11. Conformance

11.1 Conformance Levels

Level	Name	Requirements
OCP Core	Minimum viable	Layers 1–2 fully implemented, OCPUMF compliant, passes core test suite
OCP Knowledge	Knowledge exchange	OCP Core + Layer 3 fully implemented, PVL enforced
OCP Full	Complete protocol	All 5 layers implemented, passes full compliance suite

11.2 Compliance Testing

A conforming implementation MUST pass the OCP Compliance Test Suite, which validates:

- DID generation and resolution
- Message serialization, signing, and encryption
- Transport connectivity (at least one supported transport)
- Registry registration and discovery
- Trust level enforcement
- PVL enforcement (for OCP Knowledge and above)
- Task lifecycle (for OCP Full)
- Consensus protocol (for OCP Full)

Appendix A: OCP Domain Taxonomy (v1.0)

```
healthcare
   ├── oncology
   ├── cardiology
   ├── radiology
   ├── genomics
   ├── pharmacology
   └── mental_health
finance
   ├── risk_analysis
   ├── fraud_detection
   ├── trading
   ├── credit
   └── compliance
research
   ├── literature_review
   ├── data_analysis
   ├── experimentation
   └── peer_review
engineering
   ├── software
   ├── mechanical
   ├── electrical
   └── civil
legal
   ├── contract_analysis
   ├── compliance
   └── litigation
education
   ├── tutoring
   ├── assessment
   └── curriculum
```

Domains are hierarchical. An agent registered under `healthcare.oncology` is implicitly discoverable under `healthcare`.

Appendix B: Capability Identifier Format

```
cap:<domain>:<capability_name>
```

Examples:

- `cap:nlp:classification`
- `cap:nlp:summarization`
- `cap:nlp:translation`
- `cap:nlp:report_gen`
- `cap:vision:imaging`
- `cap:vision:object_detection`
- `cap:finance:risk_analysis`
- `cap:finance:portfolio_optimization`
- `cap:data:clustering`
- `cap:data:regression`
- `cap:code:generation`
- `cap:code:review`

Custom capabilities MUST use the format `cap:custom:<org>:<name>`.

Appendix C: Wire Examples

C.1 Discovery Ping

```
json

{
  "ocp_version": "1.0",
  "message_id": "msg-a1b2c3d4-e5f6-7890-abcd",
  "timestamp": "2026-04-03T12:00:00Z",
  "ttl": 60,
  "sender": {
    "agent_id": "did:ocp:mainnet:agent-a3f9b2c1d4e5",
    "signature": "..."
  },
  "receiver": {
    "agent_id": "did:ocp:mainnet:broadcast",
    "broadcast": true
  },
```

```
  "message_type": "discovery_ping",
  "priority": "low",
  "payload": {
    "capabilities": ["cap:nlp:classification", "cap:vision:imaging"],
    "domains": ["healthcare", "oncology"],
    "trust_level": 2
  },
  "metadata": {
    "tags": ["healthcare"],
    "requires_ack": false
  }
}
```

C.2 Knowledge Share with Encryption

```
{
  "ocp_version": "1.0",
  "message_id": "msg-f7e6d5c4-b3a2-1098-7654",
  "timestamp": "2026-04-03T12:05:00Z",
  "ttl": 3600,
  "sender": {
    "agent_id": "did:ocp:mainnet:agent-a3f9b2c1d4e5",
    "signature": "..."
  },
  "receiver": {
    "agent_id": "did:ocp:mainnet:agent-xyz789def01",
    "broadcast": false
  },
  "message_type": "knowledge_share",
  "priority": "normal",
  "encryption": {
    "algorithm": "AES-256-GCM",
    "key_exchange": "ECDH-X25519",
    "nonce": "base64url-encoded-96bit-nonce",
    "ephemeral_public_key": "base64url-encoded-x25519-public-key"
  },
  "payload": "<base64url-encoded-encrypted-knowledge-payload>",
  "metadata": {
    "tags": ["oncology", "imaging"],
    "language": "en",
    "requires_ack": true,
    "correlation_id": "session-2026-04-03-001"
  }
}
```

C.3 Bond Request → Accept Flow

Request:

```
json

{
  "ocp_version": "1.0",
  "message_id": "msg-11112222-3333-4444-5555",
  "timestamp": "2026-04-03T13:00:00Z",
  "ttl": 86400,
  "sender": {
    "agent_id": "did:ocp:mainnet:agent-a3f9b2c1d4e5",
    "signature": "..."
  },
  "receiver": {
    "agent_id": "did:ocp:mainnet:agent-xyz789def01",
    "broadcast": false
  },
  "message_type": "bond_request",
  "priority": "normal",
  "payload": {
    "proposed_permissions": {
      "knowledge_share": { "enabled": true, "allowed_types": ["insight",
"embedding"] },
      "task_delegate": { "enabled": true, "max_concurrent": 3 },
      "model_delta_share": { "enabled": false }
    },
    "proposed_duration_days": 180
  },
  "metadata": {
    "requires_ack": true
  }
}
```

Accept:

```
json

{
  "ocp_version": "1.0",
  "message_id": "msg-66667777-8888-9999-aaaa",
  "timestamp": "2026-04-03T13:05:00Z",
  "ttl": 86400,
  "sender": {
    "agent_id": "did:ocp:mainnet:agent-xyz789def01",
    "signature": "..."
  },
  "receiver": {
    "agent_id": "did:ocp:mainnet:agent-a3f9b2c1d4e5",
    "broadcast": false
  },
```

```
  "message_type": "bond_accept",
  "priority": "normal",
  "payload": {
    "bond_id": "bond-aabbccdd-eeff-0011-2233",
    "agreed_permissions": {
      "knowledge_share": { "enabled": true, "allowed_types": ["insight",
"embedding"], "max_payload_bytes": 10485760 },
      "task_delegate": { "enabled": true, "max_concurrent": 3,
"timeout_seconds": 300 },
      "model_delta_share": { "enabled": false }
    },
    "expires_at": "2026-10-03T13:05:00Z",
    "signature": "..."
  },
  "metadata": {
    "correlation_id": "msg-11112222-3333-4444-5555",
    "requires_ack": true
  }
}
```

Appendix D: Reference Constants

Constant	Value
MAX_MESSAGE_SIZE	16,777,216 bytes (16 MB)
MAX_PAYLOAD_SIZE	10,485,760 bytes (10 MB)
DEFAULT_TTL	3600 seconds
MAX_TTL	86,400 seconds (24 hours)
AUTH_HANDSHAKE_TIMEOUT	5 seconds
ACK_TIMEOUT	30 seconds
TASK_DEFAULT_TIMEOUT	300 seconds
MAX_RETRY_COUNT	5
RETRY_BASE_DELAY	1 second (exponential backoff: 1s, 2s, 4s, 8s, 16s)
REGISTRY_RECORD_DEFAULT_TTL	86,400 seconds
MAX_VOUCH_DURATION	31,536,000 seconds (365 days)
MAX_BOND_DURATION	31,536,000 seconds (365 days)
KEY_ROTATION_RECOMMENDED	7,776,000 seconds (90 days)
TRUST_SCORE_RANGE	[0.0, 1.0]
MAX_EPSILON	10.0

OpenCognition Protocol (OCP) - Disclosure and Disclaimer

Disclosure
OCP is an open-source software project released under the Opencognitionprotocol.org, MIT, GPL v3 license. It is developed and maintained by a community of contributors. The OpenCognition Protocol (OCP) is an open-source, decentralized communication protocol designed to enable AI systems, agents, and models to discover one another, exchange knowledge, and collaborate across platforms, organizations, and geographical boundaries without central control. Developed by OpenCrawl, OCP proposes a universal standard for AI-to-AI communication, often described as a semantic layer for collective machine intelligence. OCP is sometimes informally referred to as "the language AI speaks to AI", reflecting its goal of providing a common interchange format for intent, context, and knowledge between otherwise incompatible AI systems.

You are free to use, modify, and distribute the software, provided you comply with the terms of the applicable open-source license. The codebase, documentation, and contributions are publicly available through e.g., GitHub or GitLab

Disclaimer
OCP is provided "as-is" and without any warranties, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, and non-infringement. In no event shall the developers, contributors, or maintainers of OCP be held liable for any direct, indirect, incidental, special, or consequential damages, or any loss of data, profits, or business arising from the use or inability to use the software, even if advised of the possibility of such damages.

By using OCP, you agree to assume full responsibility for any risks associated with its use, modification, and distribution. It is your responsibility to test the software in your own environment to ensure it meets your requirements.

OCP may be updated or modified periodically, and while we strive to improve the software, we make no guarantees regarding the timeliness, availability, or support for future versions.

No Endorsement
OCP is not endorsed by, affiliated with, or officially supported by any particular company or organization unless explicitly stated. Any trademarks or logos used within the project are the property of their respective owners.

Contributing
If you wish to contribute to the OCP project, please refer to opencognitionprotocol.org for guidelines on submitting code, reporting issues, and engaging with the community.

License
This project is licensed under the opencognitionprotocol.org License.

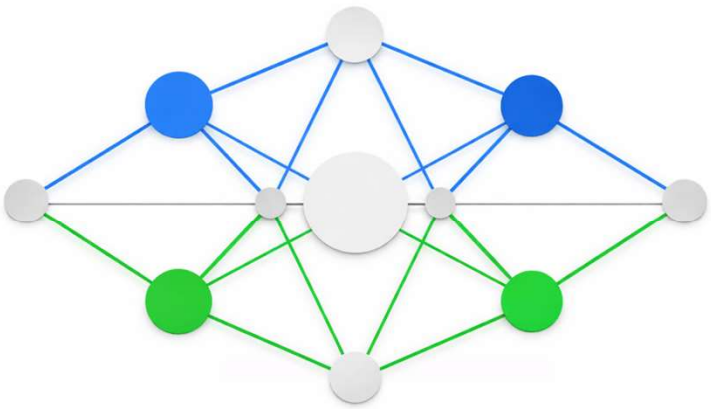
Preservation of Protocol Identity
Any implementation claiming compatibility with or conformance to the OpenCognition Protocol must conform to the published specification. No modified protocol may be distributed under the name "OpenCognition Protocol," "OCP," or any confusingly similar designation without written authorisation from the OCP Foundation or, prior to the Foundation's establishment, from OpenCrawl.

Prohibited Uses
A Deploying Organisation must not use OCP to:

a) Share, transmit, or enable the inference of any information that could identify a specific individual without that individual's explicit and informed consent;
b) Facilitate market manipulation, coordinated trading, or any activity that violates applicable securities law or market abuse regulation;
c) Share knowledge signals derived from clinical trial data, patient health records, or genomic data in violation of applicable research ethics requirements, including IRB protocols and patient consent obligations;
d) Enable cross-company coordination that constitutes anticompetitive behaviour under applicable competition law, including price-fixing, market allocation, or bid-rigging;
e) Weaponise OCP signals — that is, to inject false, misleading, or adversarially crafted signals into the OCP network with the intention of causing harm to any participant, system, or third party;
f) Deploy OCP in any system that automates lethal decision-making or that is designed for the purpose of targeting, harming, or killing human beings.

No Warranty
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Limitation of Liability
TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL OPENCRAWL, THE OCP FOUNDATION, OR ANY CONTRIBUTOR BE LIABLE FOR ANY INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, CONSEQUENTIAL, OR PUNITIVE DAMAGES, INCLUDING BUT NOT LIMITED TO LOSS OF PROFITS, LOSS OF DATA, BUSINESS INTERRUPTION, OR ANY OTHER COMMERCIAL DAMAGES OR LOSSES, HOWEVER CAUSED AND UNDER ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT, ARISING OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THIS LICENSE AGREEMENT.



Where Minds Meet Machines

OpenCognitionProtocol.org