# AgentNet Standards

### Version 1.0.0

#### © 2025 AgentNet. All Rights Reserved.

## Table of Contents

1. AgentNet Standards – Section 1.0: Introduction and Scope
2. Admin Account Management
3. AgentNet Standards – Section 2.0: Terminology and Context
4. AgentNet Standards – Section 3.0: Architectural Overview
5. AgentNet Standards – Section 4.0: Persistent Node Identity
6. AgentNet Standards – Section 5.0: Node Registration and Verification
7. AgentNet Standards – Section 6.0: Resolver Architecture and Trust Model
8. AgentNet Standards – Section 7.0: Capsule Standards and Data Compliance
9. AgentNet Standards – Section 8.0: Federation and Inter-Resolver Protocols
10. AgentNet Standards – Section 9.0: Governance and Compliance Framework
11. AgentNet Standards – Section 10.0: Security, Privacy, and Data Protection
12. AgentNet Standards – Section 11.0: Future Extensions and Interoperability
13. AgentNet Standards - Section 12.0: Resolver Federation Framework

## Section 0: AgentNet Standards – Section 1.0: Introduction and Scope

Version 1.0.0

## Section 1.1 – Purpose

The AgentNet Standards (ANS) define the foundational architecture, governance, and operational principles of the Agentic Web — a distributed, machine-readable ecosystem of nodes, capsules, and resolvers.  
ANS establishes a universal protocol for how autonomous agents discover, trust, and interact with structured data online.

## Section 1.2 – Objectives

* Enable persistent, verifiable machine-to-machine communication.
* Define global interoperability standards for node identity, capsule data, and resolver federation.
* Ensure transparent and auditable agentic operations under clear governance.
* Support both centralized and decentralized implementations through open protocols.
* Promote ethical AI and data practices within the emerging agentic economy.

## Section 1.3 – Scope

The AgentNet Standards apply to:

* Nodes registered under any AgentNet-compliant resolver.
* Capsules and JSON-LD schemas designed for agentic discovery and exchange.
* Resolvers participating in the AgentNet federation or trust network.
* Developers and organizations building compliant agentic applications.

Excluded from the scope:

* User interface or front-end design.
* Proprietary monetization models outside the protocol layer.
* Closed, non-verifiable private networks.

## Section 1.4 – Audience

ANS is intended for:

* Developers and data architects building structured agentic systems.
* Resolver operators and registrars.
* Institutional and enterprise adopters seeking interoperability.
* Academic and standards organizations studying the agentic web.

## Section 1.5 – Document Structure

This standard is organized into eleven sections:

1. Introduction and Scope
2. Terminology and Context
3. Architectural Overview
4. Persistent Node Identity
5. Node Registration and Verification
6. Resolver Architecture and Trust Model
7. Capsule Standards and Data Compliance
8. Federation and Inter-Resolver Protocols
9. Agent Interaction and Query Protocols
10. Governance, Security, and Ethical Framework
11. Future Extensions and Evolution Path

## Section 1.6 – Summary Principle

AgentNet Standards define the framework by which intelligent systems can identify, verify, and communicate across a unified, ethical, and decentralized data fabric.

## Section 1: Admin Account Management

Version 1.0.0

# AgentNet Standard: Admin Account Management

**Version:** 1.0  
**Updated:** 05-Oct-2025  
**Status:** Published

## 1. Purpose

This standard defines the temporary administrative method for creating and managing named admin accounts within the AgentNet platform. Until a protected in-app interface is available, new admins shall be created using authenticated CLI commands directed at the backend API.

## 2. Endpoint Overview

New admin accounts are created through the existing backend route:

POST /api/admin/register

This endpoint accepts an email address and password, then creates a corresponding entry in the AdminUsers table with a hashed password and admin role.

## 3. Requirements

* The backend server must be running and reachable (default: <http://localhost:5174>).
* The operator must have terminal access to issue curl commands.
* The account being added must use a unique email address.

## 4. Command Syntax

Use the following curl command to create a new admin:

curl -X POST http://localhost:5174/api/admin/register \ -H "Content-Type: application/json" \ -d '{"email":"newadmin@agentnet.ai","password":"ChooseAStrongPassword"}'

**Response Example:**

{ "ok": true, "id": 2, "email": "newadmin@agentnet.ai" }

## 5. Verification

After the request succeeds, confirm creation with:

mysql -u root -p -e "USE agentnet\_ai; SELECT id, email, role FROM AdminUsers;"

**Expected output:**

+----+-----------------------+--------+ | id | email | role | +----+-----------------------+--------+ | 1 | steve@agentnet.ai | admin | | 2 | newadmin@agentnet.ai | admin | +----+-----------------------+--------+

## 6. Security Notes

* This route is **temporarily unauthenticated** to enable bootstrapping.
* Once multiple admins exist, the /api/admin/register route shall be secured behind authenticateToken and requireRole('admin') middleware.
* All admins must use strong, unique passwords.

### ✅ End of Standard

## Section 2: AgentNet Standards – Section 2.0: Terminology and Context

Version 1.0.0

## Section 2.1 – Purpose

This section defines the core terminology used across all AgentNet Standards to ensure consistent understanding among implementers and participants.

## Section 2.2 – Key Terms

* **AgentNet** — The global network of resolvers, nodes, and capsules governed under ANS.
* **Node** — A registered entity (organization, dataset, service, or person) represented by one or more capsules.
* **Capsule** — A structured JSON-LD object containing machine-readable data about a node, product, event, or service.
* **Resolver** — A trusted service that manages node registration, verification, discovery, and trust scoring.
* **Agent** — An autonomous system, AI, or software that consumes or contributes data to AgentNet.
* **Trust Fabric** — The cryptographic and governance infrastructure ensuring authenticity, integrity, and reliability across the network.
* **Federation** — Cooperative synchronization of multiple resolvers under shared trust and protocol standards.
* **RRA (Root Resolver Authority)** — The governing body maintaining the global root ledger, resolver certificates, and compliance standards.
* **DID (Decentralized Identifier)** — A self-sovereign cryptographic identity standard enabling independent verification.
* **ANS (AgentNet Standards)** — The codified framework defining operational, architectural, and ethical principles for the Agentic Web.

## Section 2.3 – Related Standards and Technologies

AgentNet builds on open, widely adopted technologies:

* JSON-LD — Linked Data representation standard.
* W3C DID Core — Decentralized Identifier specification.
* Schema.org — Semantic entity definitions.
* TLS 1.3+ — Secure transport for resolvers and agents.
* OAuth 2.0 — Standardized agent authentication and authorization.

## Section 2.4 – Abbreviations

* **ANS** — AgentNet Standards
* **RRA** — Root Resolver Authority
* **DID** — Decentralized Identifier
* **JSON-LD** — JavaScript Object Notation for Linked Data
* **API** — Application Programming Interface

## Section 2.5 – Summary Principle

Clear, shared terminology ensures every participant in the Agentic Web speaks the same structural and semantic language.

## Section 3: AgentNet Standards – Section 3.0: Architectural Overview

Version 1.0.0

## Section 3.1 – Purpose

This section provides a conceptual overview of the AgentNet architecture, defining the layers and functional relationships among nodes, resolvers, capsules, and agents.

## Section 3.2 – AgentNet Layer Model

AgentNet architecture is organized into five functional layers:

* **Identity Layer** — Establishes unique, verifiable identifiers for nodes via resolver-issued or decentralized IDs.
* **Data Layer** — Defines structured capsules using JSON-LD for interoperability and semantic clarity.
* **Trust Layer** — Ensures cryptographic integrity, validation, and transparency through resolvers and federation.
* **Interaction Layer** — Enables queries, subscriptions, and transactions between agents and nodes.
* **Governance Layer** — Maintains accountability, ethics, and standardization across the ecosystem.

## Section 3.3 – System Components

Each component in the AgentNet architecture plays a defined role:

* **Nodes** — Publish and maintain capsules describing entities, products, or datasets.
* **Resolvers** — Manage registry information, verify ownership, and synchronize state across the federation.
* **Agents** — Perform discovery, evaluation, and communication between nodes.
* **Federation Network** — Maintains distributed consensus and ledger integrity across resolvers.
* **Governance Authorities** — Define compliance standards, certification, and ethical oversight.

## Section 3.4 – Data Flow Overview

A standard AgentNet data flow proceeds as follows:

1. A node is registered with a resolver and receives a permanent identifier.
2. The node’s capsule(s) are validated for compliance and stored in the resolver registry.
3. Agents query resolvers to locate relevant nodes or capsules.
4. Resolvers return trusted results including verification metadata.
5. Updates, ownership transfers, and revocations propagate through the federation ledger.

## Section 3.5 – Interoperability and Extensibility

AgentNet is designed for compatibility with established open standards and for long-term scalability:

* Interoperable with W3C, Schema.org, and JSON-LD ecosystems.
* API-accessible for integration with other knowledge networks.
* Versioned schema contexts enable forward-compatible evolution.
* Modular structure allows localized or domain-specific extensions.

## Section 3.6 – Summary Principle

AgentNet’s layered architecture ensures modularity, resilience, and transparency — forming the foundation of a verifiable, machine-readable Internet.

## Section 4: AgentNet Standards – Section 4.0: Persistent Node Identity

Version 1.0.0

## Section 4.1 – Purpose

AgentNet must guarantee long-term continuity and resolvability of nodes even when their human-facing websites, domains, or hosting arrangements change.  
This section defines a layered identity model that supports domain-linked, resolver-anchored, and decentralized (DID-based) identifiers.

## Section 4.2 – Core Principle

Domain linkage enhances trust but must not be a dependency for machine resolvability.  
Every node must possess at least one permanent resolver-issued identifier that remains valid regardless of domain status.

## Section 4.3 – Identity Tiers

AgentNet defines three primary tiers of node identity:

* **Tier 1 – Domain-Linked ID**  
  Example: https://example.com
  + Provides human-readable linkage to an existing website or DNS identity.
  + Valid only while the domain is active.
* **Tier 2 – Resolver-Anchored ID**  
  Example: agentnet://node/abcd1234
  + The primary machine-facing identifier issued at node registration.
  + Guaranteed persistent by the resolver regardless of domain status.
* **Tier 3 – Decentralized (DID) ID**  
  Example: did:agentnet:ef5678...
  + A self-signed, cryptographic identifier enabling off-network verification and ownership proofs.
  + Used primarily for high-assurance or self-hosted nodes.

## Section 4.4 – Recommended Capsule Schema

Example JSON-LD representation of persistent node identity:

{ "@context": "https://agentnet.ai/context", "@id": "agentnet://node/abcd1234", "agentnet:domain": "example.com", "agentnet:did": "did:agentnet:ef5678...", "agentnet:owner": "Example Web Co.", "agentnet:resolver": "https://resolver.agentnet.ai", "agentnet:domainStatus": "active" }

If a domain expires, resolvers update "agentnet:domainStatus": "expired" but continue to honor permanent identifiers.

## Section 4.5 – Resolver Behavior and Lifecycle Handling

Resolvers maintain continuity of node identity regardless of external namespace changes.

### Domain Expiration Handling

1. Verify domain status periodically.
2. Update node metadata with "agentnet:domainStatus": "expired".
3. Continue resolving via permanent IDs.
4. Notify node owner through registered webhook or contact address.
5. Allow optional redirect using "agentnet:redirectDomain".

**Example metadata update**

{ "@id": "agentnet://node/abcd1234", "agentnet:domainStatus": "expired", "agentnet:redirectDomain": "newsite.com" }

### Ownership Persistence

Ownership is bound to resolver-registered credentials or cryptographic keys, not to domain registrars.

### Cryptographic Validation (Tier 3)

Nodes without domains must include a signed ownership record:

{ "agentnet:signature": "0xabc123...", "agentnet:publicKey": "0xdef456..." }

### Node Reactivation

If a domain is renewed or replaced, the resolver updates status to "active" and retains full ledger continuity.

### Expired Node Policy

Nodes are never deleted solely for domain expiration.  
Deletion occurs only by verified owner request or long-term inactivity under governance policy.

## Section 4.6 – Summary Principle

AgentNet resolvers must preserve continuity of node identity and accessibility independent of any external namespace, ensuring persistent machine interoperability across all future network states.

## Section 5: AgentNet Standards – Section 5.0: Node Registration and Verification

Version 1.0.0

## Section 5.1 – Purpose

This section defines the process by which new nodes are registered, verified, and published into the AgentNet ecosystem.  
Registration ensures that each node receives a unique and persistent identifier while meeting baseline authenticity, schema, and ownership criteria.

## Section 5.2 – Registration Principles

* Each resolver serves as an official registrar for the AgentNet network.
* A node cannot exist without at least one verified resolver registration record.
* Registration events are cryptographically signed and recorded in the resolver’s ledger.
* Verification must confirm both technical validity (schema compliance) and identity authenticity (ownership credentials).
* All submissions use secure transport (HTTPS / TLS 1.3+).

## Section 5.3 – Registration Workflow

The following steps describe the full lifecycle of a new node registration:

1. **Submission** – The node owner submits a registration request to a resolver’s public API.
2. **Validation** – The resolver validates JSON-LD schema structure and mandatory fields.
3. **Ownership Check** – Resolver verifies the requester’s email, domain, or DID signature.
4. **Resolver Assignment** – Resolver issues a permanent agentnet://node/ identifier.
5. **Verification Token** – A temporary token is returned to confirm identity ownership.
6. **Ledger Entry** – Once verified, the resolver adds the node to the federation ledger.
7. **Publication** – The node becomes queryable and discoverable across federated resolvers.

## Section 5.4 – Example Registration Request

Example JSON body submitted by a node owner during registration:

{ "@context": "https://agentnet.ai/context", "agentnet:action": "registerNode", "agentnet:owner": { "name": "Example Web Co.", "email": "admin@example.com", "organizationType": "LLC" }, "agentnet:domain": "example.com", "agentnet:did": "did:agentnet:ef5678abcd", "agentnet:publicKey": "0xDEF456789ABC", "agentnet:timestamp": "2025-10-05T18:30:00Z", "agentnet:capsuleSummary": "Primary dataset and service description for Example Web Co." }

## Section 5.5 – Resolver Verification Response

After successful validation, the resolver returns a verification response containing registration data and a verification token:

{ "@context": "https://agentnet.ai/context", "@type": "agentnet:NodeVerificationResponse", "agentnet:status": "pendingVerification", "agentnet:nodeId": "agentnet://node/abcd1234", "agentnet:resolver": "https://resolver.agentnet.ai", "agentnet:verificationToken": "b7f1c4aa-112f-4e73-912b-9de8f4f8b11a", "agentnet:expires": "2025-10-06T18:30:00Z", "agentnet:message": "A verification link has been sent to admin@example.com." }

## Section 5.6 – Ownership Verification Completion

The node owner must complete verification by one of the following methods:

* **Email Link** – Clicking the verification URL provided in the resolver email.
* **DNS Record** – Adding a TXT record with the provided verification token.
* **DID Signature** – Submitting a signed payload proving ownership of the DID key.

Example signed verification submission:

{ "@context": "https://agentnet.ai/context", "agentnet:action": "verifyNodeOwnership", "agentnet:nodeId": "agentnet://node/abcd1234", "agentnet:verificationToken": "b7f1c4aa-112f-4e73-912b-9de8f4f8b11a", "agentnet:signature": "0xA3B6C9E7D9F12AB34...", "agentnet:publicKey": "0xDEF456789ABC", "agentnet:timestamp": "2025-10-05T18:42:10Z" }

## Section 5.7 – Final Confirmation and Publication

Upon successful verification, the resolver finalizes registration and returns confirmation:

{ "@context": "https://agentnet.ai/context", "@type": "agentnet:NodeRegistrationComplete", "agentnet:status": "verified", "agentnet:nodeId": "agentnet://node/abcd1234", "agentnet:resolver": "https://resolver.agentnet.ai", "agentnet:owner": "Example Web Co.", "agentnet:verifiedAt": "2025-10-05T18:43:00Z", "agentnet:ledgerHash": "0xa84f6b2cbe981ff...", "agentnet:message": "Node successfully verified and published to the federation ledger." }

## Section 5.8 – Error Handling and Rejection Cases

Resolvers must handle invalid or malicious submissions gracefully.

Typical error responses include:

{ "@context": "https://agentnet.ai/context", "@type": "agentnet:ErrorResponse", "agentnet:status": "rejected", "agentnet:errorCode": "E-REG-INVALID-DOMAIN", "agentnet:message": "Domain ownership could not be verified. Please add the TXT record and retry.", "agentnet:timestamp": "2025-10-05T18:35:00Z" }

Resolvers must log and throttle repeated failed attempts to protect against abuse.

## Section 5.9 – Security and Compliance Notes

* All resolver communications must use TLS 1.3 or newer.
* Verification tokens are single-use and expire within 24 hours.
* All ledger entries must include a cryptographic hash of the registration record.
* Personally identifiable information (PII) must not be published in public capsules.
* Federation members must maintain audit logs of verification events for a minimum of 12 months.

## Section 5.10 – Summary Principle

Node registration and verification form the foundation of AgentNet trust.  
Resolvers must ensure that every node admitted to the network is uniquely identified, verifiably owned, and securely recorded for federated trust and discovery.

## Section 6: AgentNet Standards – Section 6.0: Resolver Architecture and Trust Model

Version 1.0.0

## Section 6.1 – Purpose

Resolvers are the backbone of AgentNet’s decentralized trust and discovery system.  
They maintain node registries, enforce identity persistence, validate capsule authenticity, and synchronize trust state across the federation.  
This section defines the architectural roles, trust layers, and inter-resolver synchronization protocols governing the AgentNet network.

## Section 6.2 – Core Responsibilities of a Resolver

* Maintain a verifiable ledger of node identities and status.
* Perform registration, verification, and periodic revalidation.
* Resolve any agentnet://node/ or did:agentnet: identifier to the correct capsule endpoint.
* Synchronize trust data with peer resolvers through a federated consensus protocol.
* Enforce rate limiting, abuse prevention, and identity integrity.
* Issue, validate, and revoke resolver certificates as directed by the Root Resolver Authority (RRA).

## Section 6.3 – Architectural Overview

Each resolver operates through the following internal layers:

1. **Ingress Layer**
   * Handles API traffic (HTTPS, WebSocket, or gRPC).
   * Performs authentication and rate limiting.
2. **Validation Layer**
   * Parses JSON-LD payloads.
   * Validates required fields and schema structure.
   * Verifies signatures and timestamps.
3. **Ledger Layer**
   * Maintains a tamper-resistant record of node registrations.
   * Uses cryptographic hashes for change detection.
   * Supports query operations for lookup and audit.
4. **Federation Layer**
   * Syncs changes with other resolvers using signed update bundles.
   * Confirms cross-resolver consistency via trust consensus.
   * Reports state to the RRA for periodic audits.
5. **API Layer**
   * Provides endpoints for public lookup, registration, verification, and health status.
   * Returns structured responses in JSON-LD.

## Section 6.4 – Example Resolver Metadata Record

Each resolver publishes a public metadata record describing its capabilities and configuration:

{ "@context": "https://agentnet.ai/context", "@type": "agentnet:ResolverMetadata", "agentnet:resolverId": "agentnet://resolver/resolver-001", "agentnet:name": "Primary North America Resolver", "agentnet:url": "https://resolver.agentnet.ai", "agentnet:publicKey": "0x001A2B3C4D5E...", "agentnet:certificate": "RRA-CERT-2025-NA-01", "agentnet:version": "1.0.0", "agentnet:lastSync": "2025-10-05T18:50:00Z", "agentnet:status": "active", "agentnet:capabilities": [ "registerNode", "verifyNodeOwnership", "resolveNode", "federationSync" ] }

## Section 6.5 – Federation and Trust Synchronization

AgentNet employs a Federated Trust Mesh model for resolver coordination.

### Synchronization Process

1. **Change Detection** – Each resolver monitors ledger deltas since the last sync.
2. **Bundle Generation** – Resolver packages updates into a signed “Trust Bundle.”
3. **Peer Exchange** – Bundles are shared among authorized peer resolvers.
4. **Verification** – Peers validate each bundle’s signature, timestamp, and hash chain.
5. **Consensus Confirmation** – Matching hashes across peers mark the bundle as verified.
6. **Ledger Update** – All peers apply verified changes to maintain a unified trust state.

Example trust bundle:

{ "@context": "https://agentnet.ai/context", "@type": "agentnet:TrustBundle", "agentnet:issuer": "agentnet://resolver/resolver-001", "agentnet:issuedAt": "2025-10-05T18:55:00Z", "agentnet:updates": [ { "agentnet:nodeId": "agentnet://node/abcd1234", "agentnet:status": "verified", "agentnet:ledgerHash": "0xa84f6b2cbe981ff..." } ], "agentnet:signature": "0xF1A2B3C4D5E678..." }

## Section 6.6 – Root Resolver Authority (RRA)

The **Root Resolver Authority (RRA)** functions as the governing certification and compliance body for resolvers.

### Responsibilities

* Issue and revoke resolver certificates.
* Maintain the root ledger and federation registry.
* Audit resolver uptime, accuracy, and security compliance.
* Define baseline operational and cryptographic standards.

### Example Root Authority Record

{ "@context": "https://agentnet.ai/context", "@type": "agentnet:RootResolverAuthority", "agentnet:authorityName": "AgentNet Global RRA", "agentnet:rootLedgerUrl": "https://root.agentnet.ai/ledger", "agentnet:certificatePolicy": "RRA-CP-1.2", "agentnet:lastAudit": "2025-09-30T00:00:00Z", "agentnet:auditor": "AgentNet Trust Council" }

## Section 6.7 – Trust Fabric and Certificates

* All resolvers must hold valid, signed RRA certificates before joining the federation.
* Certificates are short-lived (≤ 90 days) and automatically rotated.
* Federation messages must include both resolver and bundle signatures.
* Resolvers without current certificates are automatically quarantined until renewed.

## Section 6.8 – Security Requirements

* TLS 1.3 + mandatory for all network transport.
* Mutual certificate authentication (mTLS) between resolvers.
* Cryptographic algorithms must meet NIST FIPS 140-3 compliance.
* Trust bundle signatures must use ECDSA P-256 or stronger.
* Each resolver must isolate ledger storage from public-facing APIs.
* All trust data is immutable once committed.

## Section 6.9 – Federation Health Monitoring

Resolvers must publish health metrics and sync states for observability:

{ "@context": "https://agentnet.ai/context", "@type": "agentnet:ResolverHealth", "agentnet:resolverId": "agentnet://resolver/resolver-001", "agentnet:uptime": "99.991%", "agentnet:lastSyncLag": "42s", "agentnet:peersOnline": 27, "agentnet:certExpires": "2025-12-31T23:59:59Z" }

Monitoring dashboards aggregate these metrics to ensure network stability and timely federation convergence.

## Section 6.10 – Summary Principle

AgentNet resolvers collectively form a secure, distributed trust network.  
Through certificate governance, federated consensus, and cryptographic accountability, they guarantee that every node identity and capsule resolution remains authentic, verifiable, and globally synchronized across the AgentNet ecosystem.

## Section 7: AgentNet Standards – Section 7.0: Capsule Standards and Data Compliance

Version 1.0.0

## Section 7.1 – Purpose

This section defines the structure, content rules, and validation standards for **capsules**, the atomic data objects of the AgentNet ecosystem.  
Capsules are JSON-LD representations of facts, entities, products, or datasets that agents can interpret without human intervention.  
The goal is to ensure every capsule is verifiable, interoperable, and compliant with the AgentNet semantic model.

## Section 7.2 – Capsule Definition

A **capsule** is a self-contained JSON-LD document containing:

* A unique @id representing the entity or concept.
* A contextual schema reference via @context.
* A resolver or node linkage for authenticity.
* Descriptive properties, metadata, and verification data.
* Optional provenance or trust metadata.

Capsules may represent:

* Entities (companies, individuals, datasets)
* Products or services
* Events or observations
* Research or informational records

## Section 7.3 – Core Capsule Fields

All compliant capsules must include the following properties.  
Each entry lists the **field name**, **type**, and **description** for clarity and compatibility with compact renderers.

### @context

* **Type:** URL
* **Description:** Links to the AgentNet or external schema definitions, typically https://agentnet.ai/context.

### @id

* **Type:** URI
* **Description:** Globally unique identifier (resolver- or DID-based) for this capsule.

### @type

* **Type:** String
* **Description:** Capsule class or entity type (e.g., agentnet:Product, agentnet:Organization).

### agentnet:owner

* **Type:** String
* **Description:** Registered node owner’s name or unique identifier.

### agentnet:resolver

* **Type:** URI
* **Description:** Resolver endpoint responsible for registration and verification of this capsule.

### agentnet:timestamp

* **Type:** ISO 8601 String
* **Description:** UTC timestamp marking creation or last update of the capsule.

### agentnet:version

* **Type:** String
* **Description:** Semantic version number describing this capsule’s schema or structure (e.g., 1.0.0).

**Example Usage Reference:**  
See Section 7.4 for a complete JSON-LD capsule example illustrating these core fields in context.

## Section 7.4 – Example Capsule (Product Entity)

{ "@context": [ "https://agentnet.ai/context", "https://schema.org" ], "@id": "agentnet://node/abcd1234/capsule/001", "@type": "agentnet:Product", "agentnet:owner": "Example Web Co.", "agentnet:resolver": "https://resolver.agentnet.ai", "agentnet:timestamp": "2025-10-05T19:00:00Z", "agentnet:version": "1.0.0", "name": "Smart Home Controller", "description": "An IoT controller for automating home energy and lighting systems.", "brand": "Example Home", "sku": "SHC-2025", "price": 199.95, "currency": "USD", "availability": "https://schema.org/InStock", "category": "Electronics > Home Automation", "agentnet:checksum": "sha256-394efab29c9a...", "agentnet:signature": "0xA1B2C3D4E5F6...", "agentnet:proof": { "agentnet:ledgerHash": "0xF98E22B7A9...", "agentnet:verifiedBy": "agentnet://resolver/resolver-001" } }

## Section 7.5 – Validation and Canonicalization

Resolvers must perform the following validation steps before accepting or publishing a capsule:

1. Confirm required fields (@context, @id, @type, agentnet:owner, agentnet:resolver).
2. Validate JSON-LD structure and ensure all contexts resolve successfully.
3. Canonicalize the JSON object (sorted keys, no whitespace, UTF-8 encoding).
4. Compute a SHA-256 checksum and store it as agentnet:checksum.
5. Validate the capsule’s digital signature against the owner’s registered public key.
6. Record checksum and verification result in the resolver ledger.

**Canonicalization steps**

* Sort all keys lexicographically.
* Remove comments and non-semantic whitespace.
* Encode UTF-8.
* Compute SHA-256 hash.

## Section 7.6 – Capsule Update Rules

* Every update must increment the agentnet:version value.
* Updated capsules must reference the previous checksum via agentnet:previousChecksum.
* The resolver ledger must maintain a full immutable history of all prior versions.

**Example update record**

{ "@id": "agentnet://node/abcd1234/capsule/001", "agentnet:version": "1.0.1", "agentnet:previousChecksum": "sha256-394efab29c9a...", "agentnet:checksum": "sha256-832be21cc421...", "agentnet:timestamp": "2025-10-06T00:00:00Z" }

## Section 7.7 – Data Compliance Requirements

* Use open vocabularies where possible (Schema.org, Dublin Core, etc.).
* Proprietary terms must use the agentnet: namespace.
* Personally identifiable information (PII) must be anonymized or tokenized.
* Numeric values must follow SI or ISO standards.
* Dates must use ISO 8601 UTC format.
* All text must be UTF-8 encoded and free of control characters.
* Capsules should be human-readable yet optimized for agent parsing.

## Section 7.8 – Capsule Integrity Verification

To verify capsule authenticity:

1. Retrieve capsule from resolver.
2. Canonicalize and hash to recompute checksum.
3. Compare result to agentnet:checksum.
4. Validate signature using owner’s public key.
5. Cross-check ledger entry for matching hash.
6. Mark status as trusted or compromised.

**Verification record example**

{ "@context": "https://agentnet.ai/context", "@type": "agentnet:CapsuleVerificationRecord", "agentnet:capsuleId": "agentnet://node/abcd1234/capsule/001", "agentnet:verifiedAt": "2025-10-05T19:30:00Z", "agentnet:checksumValid": true, "agentnet:signatureValid": true, "agentnet:ledgerMatch": true, "agentnet:status": "trusted" }

## Section 7.9 – Error Conditions and Recovery

Resolvers must reject or quarantine non-compliant capsules.  
Typical rejection causes:

* Missing required fields.
* Invalid checksum or mismatched hash.
* Schema or JSON-LD context errors.
* Invalid or expired signature.
* Version regression or checksum collision.

**Error response example**

{ "@context": "https://agentnet.ai/context", "@type": "agentnet:ErrorResponse", "agentnet:status": "rejected", "agentnet:errorCode": "E-CAPSULE-INVALID-CHECKSUM", "agentnet:message": "Checksum does not match canonicalized payload.", "agentnet:timestamp": "2025-10-05T19:35:00Z" }

## Section 7.10 – Summary Principle

Capsules are the foundation of structured truth within AgentNet.  
By enforcing semantic consistency, cryptographic integrity, and transparent version history, AgentNet ensures all agents can exchange and reason over data with confidence and authenticity.

## Section 8: AgentNet Standards – Section 8.0: Federation and Inter-Resolver Protocols

Version 1.0.0

## Section 8.1 – Purpose

This section defines the protocols by which resolvers within the AgentNet federation communicate, synchronize, and maintain global state consistency.  
Federation ensures that node registrations, capsule updates, and trust metadata are verifiable across all participating resolvers regardless of geography or ownership.

## Section 8.2 – Federation Model Overview

AgentNet operates under a **Federated Trust Mesh** model.  
Each resolver maintains authority over its own registered nodes but synchronizes trust and status with peer resolvers through signed update bundles.

Key principles:

* No single resolver controls the network state.
* Trust is distributed and cryptographically verifiable.
* Synchronization occurs on an event-driven, delta-based model.
* The Root Resolver Authority (RRA) provides certificate governance and audit oversight.

## Section 8.3 – Synchronization Event Types

Resolvers exchange **trust bundles** containing one or more of the following event types:

* **Node Registration Events** — Creation or deletion of node identities.
* **Capsule Publication Events** — New or updated capsules with version references.
* **Verification Events** — Ownership or domain verification updates.
* **Revocation Events** — Expired or withdrawn certificates, signatures, or credentials.
* **Health Events** — Status and uptime data for federation observability.

Each event is timestamped, hashed, and signed by the issuing resolver.

## Section 8.4 – Trust Bundle Schema

A trust bundle is a cryptographically signed JSON-LD object that encapsulates recent verified changes.

{ "@context": "https://agentnet.ai/context", "@type": "agentnet:TrustBundle", "agentnet:issuer": "agentnet://resolver/resolver-001", "agentnet:issuedAt": "2025-10-05T20:00:00Z", "agentnet:events": [ { "agentnet:eventType": "NodeRegistration", "agentnet:nodeId": "agentnet://node/abcd1234", "agentnet:status": "verified", "agentnet:ledgerHash": "0x8aa0f1f9e8..." }, { "agentnet:eventType": "CapsuleUpdate", "agentnet:capsuleId": "agentnet://node/abcd1234/capsule/001", "agentnet:checksum": "sha256-832be21cc421..." } ], "agentnet:previousBundleHash": "0x999af4e01e2...", "agentnet:signature": "0xF1A2B3C4D5E678..." }

## Section 8.5 – Synchronization Protocol

1. **Delta Detection** — Resolver identifies local changes since last synchronization.
2. **Bundle Creation** — The changes are packaged into a signed trust bundle.
3. **Peer Distribution** — Bundles are sent to authorized peer resolvers via mTLS-secured API calls.
4. **Verification** — Peers validate bundle signature, timestamp, and hash chain.
5. **Consensus Merge** — When multiple resolvers confirm identical bundle hashes, the update is accepted.
6. **Ledger Commit** — Changes are recorded locally and marked as globally synchronized.

Synchronization can occur:

* Automatically (scheduled interval, e.g., every 5 minutes).
* On-demand (triggered by registration or update events).
* Manually (administrator-initiated).

## Section 8.6 – Trust Consensus Algorithm

To maintain consistency across resolvers, AgentNet uses a **Hash Consensus Protocol** (HCP) defined by:

1. Each trust bundle includes a previousBundleHash linking it to the last confirmed bundle.
2. Peers compute hash chains and verify signature continuity.
3. At least N of M peers must confirm a bundle hash match for acceptance.
4. Conflicts trigger an automated reconciliation process using RRA arbitration.

This approach provides integrity without requiring full blockchain overhead, while maintaining federated verifiability.

## Section 8.7 – Federation Authentication and Security

All resolver-to-resolver communication must comply with the following security standards:

* **Transport:** TLS 1.3 or newer with mutual authentication (mTLS).
* **Certificates:** Short-lived, RRA-issued resolver certificates.
* **Authorization:** API keys or OAuth2 tokens bound to resolver IDs.
* **Encryption:** AES-256-GCM for stored bundles; ECDSA-P256 for signatures.
* **Rate Limiting:** Prevents sync flooding or malicious replay attempts.
* **Audit Logging:** Each exchange is logged with timestamps, hashes, and peer IDs.

## Section 8.8 – Example Federation Exchange

**Outbound Bundle Request**

POST https://resolver-eu.agentnet.ai/api/v1/federation/sync { "agentnet:issuer": "agentnet://resolver/resolver-001", "agentnet:bundleHash": "0x8aa0f1f9e8...", "agentnet:timestamp": "2025-10-05T20:01:00Z", "agentnet:signature": "0xABCDEF123456..." }

**Inbound Response**

{ "@context": "https://agentnet.ai/context", "@type": "agentnet:TrustBundleAck", "agentnet:receiver": "agentnet://resolver/resolver-002", "agentnet:receivedAt": "2025-10-05T20:01:05Z", "agentnet:status": "accepted", "agentnet:verified": true, "agentnet:ledgerHash": "0xD45E9A991C..." }

## Section 8.9 – Error Handling and Retries

Resolvers must implement retry logic with exponential backoff for failed syncs.  
Common error conditions include:

* Expired or revoked resolver certificate.
* Signature mismatch on incoming trust bundle.
* Hash chain break or missing previous bundle.
* Network or timeout errors.

**Error Response Example**

{ "@context": "https://agentnet.ai/context", "@type": "agentnet:ErrorResponse", "agentnet:status": "rejected", "agentnet:errorCode": "E-FED-HASH-MISMATCH", "agentnet:message": "Trust bundle hash does not align with previous ledger state.", "agentnet:timestamp": "2025-10-05T20:05:00Z" }

## Section 8.10 – Summary Principle

The federation protocol ensures that all resolvers in the AgentNet ecosystem remain cryptographically synchronized, tamper-resistant, and mutually auditable.  
Through distributed hash consensus, short-lived certificates, and signed trust bundles, AgentNet achieves durable trust without centralization—balancing integrity, performance, and scalability across all federated participants.

## Section 9: AgentNet Standards – Section 9.0: Governance and Compliance Framework

Version 1.0.0

## Section 9.1 – Purpose

This section defines the **governance structure**, **compliance obligations**, and **oversight mechanisms** that ensure operational integrity and fairness across the AgentNet ecosystem.  
Governance applies to all resolvers, node operators, and federation participants under the supervision of the **Root Resolver Authority (RRA)**.

## Section 9.2 – Governance Philosophy

AgentNet’s governance model is built on three principles:

1. **Transparency** – All decisions, audits, and certificate issuances are publicly recorded.
2. **Accountability** – Each resolver and node owner bears full responsibility for compliance with published standards.
3. **Neutrality** – Governance bodies act solely to ensure technical integrity and trust continuity, not commercial advantage.

## Section 9.3 – Organizational Structure

### 1. Root Resolver Authority (RRA)

The RRA functions as the top-level governing entity responsible for:

* Defining cryptographic, operational, and compliance standards.
* Issuing and revoking resolver certificates.
* Maintaining the master trust ledger and audit registry.
* Overseeing the **AgentNet Trust Council (ATC)**.

### 2. AgentNet Trust Council (ATC)

The ATC is a rotating advisory board composed of representatives from verified resolver operators.  
It:

* Reviews proposed changes to the AgentNet Standards (ANS).
* Arbitrates disputes between resolvers.
* Recommends policy updates to the RRA.

### 3. Federation Members

Federation Members are certified resolver operators. They:

* Maintain local trust ledgers.
* Enforce the AgentNet Standards.
* Participate in synchronization and voting events.

### 4. Node Owners

Node owners register, maintain, and publish capsules. They:

* Must comply with content, schema, and authenticity rules.
* Retain ownership and liability for their published data.
* Can appeal governance actions via their hosting resolver.

## Section 9.4 – Policy and Standards Lifecycle

AgentNet Standards (ANS) evolve through a transparent, versioned process:

1. **Proposal (ANS-P)** – Draft standard introduced by ATC or RRA.
2. **Review (ANS-R)** – Distributed comment period open to federation members.
3. **Ratification (ANS-V)** – Final vote requiring ≥ 75% approval.
4. **Adoption (ANS-A)** – Official inclusion into the active standards corpus.
5. **Archival (ANS-X)** – Superseded versions retained for historical audit.

Each version must include:

* Version number and publication date.
* Editor(s) and originating body.
* Change summary and rationale.

## Section 9.5 – Compliance Obligations

### For Resolvers

* Maintain 99.9% uptime or better.
* Enforce registration and capsule validation policies.
* Rotate certificates within defined timeframes.
* Maintain signed audit logs for at least 12 months.
* Participate in annual compliance audits by the RRA.

### For Node Owners

* Ensure data accuracy and schema compliance.
* Use only authorized resolvers for publication.
* Protect private keys and access credentials.
* Respond to verification challenges within 72 hours.
* Immediately report any suspected compromise.

## Section 9.6 – Enforcement and Penalties

Violations of the AgentNet Standards are categorized by severity.  
Each level defines the type of violation, expected remediation period, and possible enforcement actions.

### **Level 1 — Minor Violations**

* **Type:** Metadata error, expired resolver certificate, minor formatting non-compliance.
* **Action:** Formal notice with a 7-day remediation window.
* **Escalation:** If unresolved, escalates automatically to Level 2.

### **Level 2 — Moderate Violations**

* **Type:** Invalid checksum, repeated schema violations, delayed verification responses.
* **Action:** Temporary suspension of federation synchronization privileges.
* **Reinstatement:** Requires successful re-audit or verification replay.

### **Level 3 — Severe Violations**

* **Type:** Data forgery, unauthorized signature usage, intentional publication of false data.
* **Action:** Immediate revocation of resolver certificate and publication ban.
* **Reinstatement:** Only through formal appeal and RRA approval following a full audit.

### **Level 4 — Critical Violations**

* **Type:** Malicious interference, systemic security breach, or deliberate compromise of trust data.
* **Action:** Permanent expulsion from the federation and public disclosure via RRA bulletin.
* **Follow-up:** Affected peers must purge all synchronization data linked to the offending resolver or node.

**Principle:**  
Enforcement actions must be proportionate, documented, and transparent.  
All penalties and subsequent appeals are logged in the Governance Ledger for audit and historical record.

## Section 9.7 – Audit and Certification

Resolvers undergo scheduled audits conducted by the RRA or an accredited third-party auditor.

**Audit Phases**

1. **Preparation** – Resolver provides access to logs, ledger hashes, and certificate metadata.
2. **Verification** – Auditor compares ledger hashes, trust bundles, and event integrity.
3. **Reporting** – Compliance report generated and filed with the ATC.
4. **Certification** – Passing resolvers receive renewed or upgraded certification.

**Example Audit Record**

{ "@context": "https://agentnet.ai/context", "@type": "agentnet:AuditRecord", "agentnet:resolverId": "agentnet://resolver/resolver-001", "agentnet:auditor": "AgentNet Trust Council", "agentnet:completedAt": "2025-10-05T21:00:00Z", "agentnet:complianceScore": 98.7, "agentnet:findings": [], "agentnet:certificateRenewed": true }

## Section 9.8 – Appeals Process

Resolvers or node owners may challenge governance actions using the formal **Appeal Submission Procedure**:

1. Submit written appeal within 10 business days of notice.
2. Include supporting evidence or correction plan.
3. ATC conducts an independent review and may request additional data.
4. RRA issues a binding decision within 30 days.
5. Final decisions are logged to the public audit ledger.

**Example Appeal Submission**

{ "@context": "https://agentnet.ai/context", "@type": "agentnet:GovernanceAppeal", "agentnet:submittedBy": "agentnet://node/abcd1234", "agentnet:reason": "Disputed suspension due to checksum mismatch.", "agentnet:evidenceUri": "https://resolver.agentnet.ai/audit/appeal123", "agentnet:timestamp": "2025-10-05T21:05:00Z" }

## Section 9.9 – Transparency and Public Disclosure

* All ratified standards, governance actions, and revocations are published in a public **Governance Ledger** accessible via API.
* Non-confidential audit summaries must be visible for at least 12 months.
* Disciplinary actions against resolvers or nodes must include justification and remediation timelines.
* The RRA shall provide an annual report summarizing network compliance health.

## Section 9.10 – Summary Principle

AgentNet’s governance and compliance framework ensures that the network remains fair, accountable, and trustworthy.  
Through transparent audits, clear appeal channels, and enforceable standards, the federation can evolve securely and responsibly while maintaining global machine-trust continuity.

## Section 10: AgentNet Standards – Section 10.0: Security, Privacy, and Data Protection

Version 1.0.0

## Section 10.1 – Purpose

This section defines the **security, privacy, and data protection** requirements that govern all AgentNet participants — including resolvers, node owners, and federation administrators.  
The objective is to safeguard the network from unauthorized access, maintain data integrity, and ensure global compliance with evolving privacy frameworks.

## Section 10.2 – Security Objectives

AgentNet enforces a layered security model designed to achieve:

1. **Confidentiality** – Sensitive data is protected from unauthorized disclosure.
2. **Integrity** – All data must remain complete, accurate, and tamper-evident.
3. **Availability** – Critical systems and resolvers remain operational under all conditions.
4. **Accountability** – All actions are attributable to identifiable entities through auditable records.
5. **Resilience** – Network operations must recover quickly from faults or attacks.

## Section 10.3 – Core Security Requirements

### **Identity and Access Control**

* All resolver operators must use multi-factor authentication for administrative access.
* Node registration and publication endpoints must be token-based and scoped per action.
* Public keys must be rotated at least every 180 days or upon suspicion of compromise.
* Access logs must be retained for a minimum of 12 months.

### **Transport and Encryption**

* All traffic must use TLS 1.3 or newer.
* Federation communication requires mutual TLS (mTLS).
* Data at rest must be encrypted using AES-256-GCM or stronger.
* Resolver backups must be encrypted and stored off-site securely.

### **Event Logging and Monitoring**

* Every registration, verification, or synchronization event must be logged.
* Logs must include timestamps, node identifiers, event hashes, and originating IPs.
* Logs must be cryptographically sealed and signed hourly.
* Intrusion detection systems should trigger alerts for unusual patterns.

## Section 10.4 – Privacy Principles

AgentNet adheres to data privacy standards consistent with **GDPR**, **CCPA**, and emerging global equivalents.  
Resolvers and node owners must uphold the following principles:

1. **Data Minimization** – Only essential information required for function may be collected.
2. **Purpose Limitation** – Data is used strictly for declared operational or verification purposes.
3. **Retention Control** – Expired or obsolete data must be deleted or anonymized.
4. **User Transparency** – Node owners must disclose data usage policies to human users, if applicable.
5. **Right to Removal** – Human-linked data capsules must support opt-out and deletion upon verified request.

## Section 10.5 – Data Protection Standards

### **At Rest**

* Data stores must implement encryption, access control lists, and checksum validation.
* Personal identifiers must be hashed or pseudonymized where possible.
* Sensitive data must never appear in plaintext ledger records.

### **In Transit**

* All inter-resolver and node interactions must use secure transport with authenticated endpoints.
* No unencrypted payloads may be transmitted over public networks.
* Temporary tokens or session credentials must expire automatically within 15 minutes.

### **During Processing**

* Temporary memory storage of sensitive fields must be minimized.
* Any caching mechanisms must clear upon transaction completion.
* Resolvers should implement in-memory encryption or volatile key storage where feasible.

## Section 10.6 – Incident Response and Recovery

All resolvers must maintain a documented **Incident Response Plan (IRP)** covering detection, containment, and recovery.

### **Required IRP Steps**

1. **Detection** – Automated anomaly detection and manual escalation procedures.
2. **Containment** – Isolate affected systems and revoke compromised credentials.
3. **Notification** – Inform RRA and impacted peers within 24 hours of confirmation.
4. **Forensic Analysis** – Preserve logs, ledger entries, and related metadata for audit.
5. **Remediation** – Apply security patches, revalidate integrity, and publish a recovery report.

**Example Incident Notification**

{ "@context": "https://agentnet.ai/context", "@type": "agentnet:IncidentReport", "agentnet:resolverId": "agentnet://resolver/resolver-001", "agentnet:incidentType": "UnauthorizedAccessAttempt", "agentnet:detectedAt": "2025-10-05T21:20:00Z", "agentnet:status": "contained", "agentnet:affectedNodes": ["agentnet://node/abcd1234"], "agentnet:notificationSent": "2025-10-05T21:30:00Z" }

## Section 10.7 – Key Management

* Each resolver must maintain an isolated Key Management Service (KMS).
* Private keys must never leave the secure enclave or HSM.
* Key derivation must use PBKDF2, Argon2, or equivalent secure algorithms.
* Key revocation lists (KRL) must be distributed through the federation within 24 hours of issuance.
* Backup encryption keys must be rotated separately and tracked in the audit ledger.

## Section 10.8 – Security Auditing

Resolvers must undergo **annual security audits** by the RRA or an authorized third party.  
The audit includes verification of encryption, access controls, key lifecycles, and incident management procedures.

**Example Audit Summary**

{ "@context": "https://agentnet.ai/context", "@type": "agentnet:SecurityAuditSummary", "agentnet:resolverId": "agentnet://resolver/resolver-002", "agentnet:auditDate": "2025-09-15T00:00:00Z", "agentnet:auditor": "AgentNet Trust Council", "agentnet:findings": 0, "agentnet:criticalVulnerabilities": [], "agentnet:certificateRenewed": true }

## Section 10.9 – Data Breach Reporting

Any confirmed breach involving personal or private node data must be reported to the RRA within **24 hours** of discovery.  
Reports must include the nature, scope, and remediation plan.

**Required breach fields**

* Resolver ID and contact.
* Time of detection and confirmation.
* Estimated impact scope (affected nodes or capsules).
* Containment actions taken.
* Planned mitigation timeline.
* Confirmation of post-incident audit scheduling.

## Section 10.10 – Summary Principle

AgentNet’s security and privacy framework is rooted in zero-trust principles, encryption by default, and continuous verification.  
By ensuring data confidentiality, integrity, and resilience across every layer of the federation, AgentNet guarantees trust persistence — even in the presence of failure or malicious activity.

## Section 11: AgentNet Standards – Section 11.0: Future Extensions and Interoperability

Version 1.0.0

## Section 11.1 – Purpose

This section outlines the forward-looking principles, mechanisms, and compatibility goals for extending AgentNet beyond its initial standards version.  
It ensures that all future evolutions of the platform remain interoperable with existing nodes, capsules, and resolvers while encouraging innovation across related ecosystems.

## Section 11.2 – Evolution Principles

1. **Backward Compatibility** – New protocol or schema versions must not invalidate existing capsules or resolver operations.
2. **Forward Extensibility** – Agents and resolvers should gracefully handle unknown fields or future schema additions.
3. **Interoperability First** – Integration with external linked-data, AI, and IoT ecosystems must use open standards wherever possible.
4. **Version Transparency** – Each change must be clearly versioned, timestamped, and documented in the AgentNet Standards repository.
5. **Federation Neutrality** – Extensions cannot advantage a single resolver, platform, or vendor implementation.

## Section 11.3 – Versioning Strategy

AgentNet uses **semantic versioning** for all standards, resolvers, and schema updates:

* **MAJOR** — Incompatible or structural change.
* **MINOR** — Backward-compatible feature addition.
* **PATCH** — Documentation, examples, or compliance updates.

Each public release includes:

* A changelog in /docs/standards/changelog/.
* JSON-LD schema updates under /schemas/vX.Y/.
* Signed publication notice in the Governance Ledger.

**Example Standard Version Record**

{ "@context": "https://agentnet.ai/context", "@type": "agentnet:StandardsVersionRecord", "agentnet:version": "1.2.0", "agentnet:publishedAt": "2026-01-15T00:00:00Z", "agentnet:editor": "AgentNet Trust Council", "agentnet:changelogUri": "https://agentnet.ai/docs/standards/changelog/ans-1.2.0", "agentnet:notes": "Introduced capsule batch signing and extended DID federation." }

## Section 11.4 – Schema Evolution and Context Management

JSON-LD schemas used in AgentNet must remain modular and discoverable.  
When new entity types or vocabularies are introduced:

1. The new @context URI must be registered with the RRA.
2. Backward-compatible aliases should be defined when renaming or relocating fields.
3. Deprecated properties must remain resolvable for at least one major version.
4. Migration guides must be published for developers and node owners.
5. Schemas must retain strict typing and JSON-LD compliance.

**Example Context Alias**

{ "@context": { "agentnet": "https://agentnet.ai/vocab/", "product": "agentnet:Product", "legacyProduct": "agentnet:Product" } }

## Section 11.5 – Cross-Network Integration

AgentNet aims to interoperate seamlessly with external systems:

* **Schema.org and W3C Standards** – Maintain alignment with global linked-data conventions.
* **AI/ML Agents** – Enable structured capsule ingestion for reasoning and automation.
* **IoT Networks** – Support device identity and event capsules for sensor data.
* **Academic and Research Graphs** – Facilitate machine-readable publishing of scientific datasets.
* **Commercial Platforms** – Allow capsule discovery via APIs, SDKs, and verified feeds.

**Example Interoperability Capsule**

{ "@context": ["https://agentnet.ai/context", "https://schema.org"], "@type": "agentnet:Dataset", "agentnet:linkedTo": "https://doi.org/10.1234/energy.reactor.dataset", "agentnet:resolver": "https://resolver.agentnet.ai", "agentnet:timestamp": "2025-10-05T22:00:00Z", "description": "Linked open dataset describing cold plasma reactor performance metrics." }

## Section 11.6 – Experimental Extensions

AgentNet supports controlled experimentation through **Provisional Extension Modules (PEMs)**:

* PEMs are sandboxed specifications approved by the ATC.
* They allow testing of new capsule types, resolver behaviors, or authentication schemes.
* Each PEM must declare:
  + Unique module ID (e.g., PEM-2025-02).
  + Maintainer contact and version.
  + Expected impact scope and duration.
* PEMs expire automatically within 12 months unless ratified as part of ANS.

**Example PEM Record**

{ "@context": "https://agentnet.ai/context", "@type": "agentnet:ProvisionalExtensionModule", "agentnet:moduleId": "PEM-2025-02", "agentnet:maintainer": "AgentNet Experimental Research Group", "agentnet:created": "2025-10-01T00:00:00Z", "agentnet:expires": "2026-10-01T00:00:00Z", "agentnet:description": "Experimental module for federated key exchange optimization." }

## Section 11.7 – Deprecation and Sunset Policy

When retiring features or schema components:

1. Mark deprecated fields using "@deprecated": true in context metadata.
2. Provide migration guidance in the changelog and developer notes.
3. Allow a minimum 12-month grace period before enforcement.
4. Issue RRA bulletins for major protocol sunsets.
5. Maintain archival access for historical audit purposes.

## Section 11.8 – Collaboration and Open Standards Alignment

AgentNet encourages alignment with international standardization efforts and cross-community development.  
Participation channels include:

* The **AgentNet Standards Repository** (Git-based contribution workflow).
* Public **Request for Comments (RFC)** submissions.
* Interoperability workshops with schema.org, W3C, and ISO/IEC working groups.
* Developer forums and structured feedback channels.

All proposed changes are logged via the Governance Ledger for traceability.

## Section 11.9 – Future Focus Areas

Planned areas of future development include:

* Enhanced agent-to-agent semantic negotiation protocols.
* Capsule compression and binary transport formats.
* Decentralized resolver discovery via DNS-over-JSON-LD.
* Quantum-safe cryptography integration.
* Cross-chain identity federation using DIDs and verifiable credentials.

## Section 11.10 – Summary Principle

AgentNet’s forward evolution is governed by continuity, openness, and interoperability.  
By preserving backward compatibility, encouraging transparent innovation, and maintaining strong ties to global data standards, AgentNet ensures its ecosystem remains adaptive, sustainable, and universally integrable across generations of intelligent systems.

## Section 12: AgentNet Standards - Section 12.0: Resolver Federation Framework

Version 1.0.0

## Section 12 -- Resolver Federation Framework

### 12.1 Purpose

The Resolver Federation Framework defines the protocols, trust alignment, and operational guidelines by which multiple AgentNet Resolvers interoperate within a unified global network.  
Federation ensures load balancing, redundancy, jurisdictional flexibility, and decentralized governance without compromising canonical data integrity.

### 12.2 Definitions

**Primary Resolver** --- The authoritative AgentNet root Resolver (resolver.agentnet.ai) that maintains the canonical registry of all Node and Capsule records.

**Federated Resolver** --- Any Resolver operated by a verified third party that adheres to ANS v1.1.0 protocols and synchronizes trust metadata with the Primary Resolver.

**Trust Sync** --- The signed cryptographic exchange by which a Federated Resolver verifies record integrity, timestamps, and policy compliance with the Primary Resolver.

**Resolver Registrar** --- An entity authorized to deploy and maintain one or more Federated Resolvers under a valid license or trust agreement.

### 12.3 Core Objectives

1. **Redundancy and Reliability** -- Provide geographically distributed Resolver endpoints to reduce latency and ensure uptime.\
2. **Data Integrity** -- Maintain verifiable synchronization with the canonical AgentNet registry.\
3. **Policy Diversity** -- Allow compliant Resolvers to apply local or domain-specific trust and filtering rules while preserving interoperability.\
4. **Sustainability and Economic Incentive** -- Enable licensed operators to monetize specialized query or caching services.

### 12.4 Federation Architecture

**Root Layer** -- Maintains canonical registry and signing authority. agentnet:PrimaryResolver

**Federation Layer** -- Hosts synchronized mirrors and regional caches. agentnet:FederatedResolver

**Edge Layer** -- Performs localized query optimization, caching, and SLA-based services. agentnet:EdgeResolver

Resolvers communicate via the **AgentNet Federation Protocol (AFP)** --- a lightweight JSON-LD--based exchange over HTTPS + WebSockets. AFP supports differential updates, record hashing, and mutual-TLS authentication.

### 12.5 Registration and Verification

1. **Application:** Operators submit a Resolver Registration Capsule containing legal entity info, jurisdiction, contact, and public key.\
2. **Evaluation:** AgentNet verifies compliance with the ANS technical and governance criteria.\
3. **Assignment:** Upon approval, a unique Resolver ID (resolver:uuid) is issued and added to the public Federation Directory.\
4. **Sync Handshake:** The Federated Resolver performs an initial Trust Sync to import baseline hashes of the canonical registry.

### 12.6 Operational Requirements

* **Uptime SLA:** ≥ 99.5 % monthly availability.\
* **Sync Interval:** Minimum one sync per 24 hours with the Primary Resolver (or per AFP negotiation).\
* **Audit Logging:** All query and update events must retain signed hashes for ≥ 90 days.\
* **Protocol Compliance:** Must implement ANS v1.1.0 AFP and TLS 1.3+ with mutual authentication.\
* **Data Residency:** Local caching must comply with jurisdictional data laws.\
* **Termination:** Non-compliant resolvers are revoked via signed Trust Revocation Notice.

### 12.7 Governance and Economics

1. **Licensing:** Federated Resolvers operate under a renewable AgentNet Resolver License.\
2. **Revenue Sharing:** Optional program whereby a percentage of paid query or SLA fees are remitted to the AgentNet Registry.\
3. **Trust Tiering:** Resolvers may be categorized as Public, Private, or Restricted based on scope and verification level.\
4. **Dispute Resolution:** Conflicts regarding record validity or trust revocation are escalated through the AgentNet Governance Board (AGB).

### 12.8 Policy Extensibility

Federated Resolvers may define local policy modules expressed in JSON-LD under agentnet:policyExtension.  
Examples include:

* Domain-specific trust filters (e.g., health, finance)\
* Query priority rules (based on velocity or reputation scores)\
* Region-specific data retention policies

All extensions must be declared in the Resolver Registration Capsule for discovery and auditing.

### 12.9 Security and Integrity

1. **Signing:** All capsule transactions and resolver updates must be signed with registered private keys.\
2. **Revocation:** Compromised or expired keys invalidate resolver trust until renewal.\
3. **Encryption:** End-to-end TLS encryption is mandatory for all Federation traffic.\
4. **Audit API:** Primary Resolver exposes a read-only endpoint for public verification of resolver status and integrity hashes.

### 12.10 Compliance Matrix

**Trust Sync** --- Mandatory · Canonical data consistency  
**Local Caching** --- Mandatory · Required for latency reduction  
**Policy Extensions** --- Optional · Must declare namespace  
**Revenue Participation** --- Optional · For commercial resolvers  
**Private Federation Mode** --- Optional · Internal enterprise networks

### 12.11 Version and Future Considerations

This framework establishes the baseline for federated operation within AgentNet v1.1.0.  
Future revisions may introduce:

* Decentralized Trust Graph attestations\
* Blockchain-anchored audit hashes\
* Autonomous resolver discovery via AgentNet AI Index\
* Cross-federation reputation and velocity scoring

### 12.12 References

* AgentNet Federation Protocol Specification (AFP v1.1.0)\
* AgentNet Governance Board Policy Manual (AGB-PM-v1.0)\
* AgentNet Security and Integrity Standard (ASIS v1.0)